The Application of Graph Theory to the Synthesis of Protocol Converters via the Interface Equation

Thesis

UNRESTRICTED

The Application of Graph Theory to the Synthesis of Protocol Converters via the Interface Equation

PhD Thesis

Alan David Pengelly

23rd January 1995

Department of Computer Science

Faculty of Mathematics

Open University

Date of submission: 20 February 1995
Date of award: 7 September 1995

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

This thesis is dedicated to Karen my wife, Christopher my son and Jessica my daughter

Contents

0.1 Publications ix

0.2 Abstract x

1 Introduction **1**

2 Formal Methods and Telecommunications **6**

2.1 Software Engineering and Formality 6

2.2 Outline of Formal Methods 9

2.3 Formal Methods and Concurrency 12

2.4 Applying Formal Methods to Telecommunications 16

2.5 Summary 18

3 Review of Formal Methods in Protocol Engineering **20**

3.1 Introduction 21

3.2 Communication Systems Development 22

3.3 Formal Methods 23

3.4 Protocol engineering 25

3.5 The Service Specification 26

3.6 Protocol Specification 29

3.7 Protocol Analysis 36

3.8 Protocol Conversion 49

3.9 Summary 59

4 The Interface Equation **64**

4.1 Introduction 64

4.2 Solving the interface equation 66

4.3 The discarding and constructive algorithms 68

4.4 A small example 78

4.5 Conclusion 87

5 A Graph Theoretic Approach **91**

5.1 Introduction 92

5.2 Quotients in computer science 93

5.3 A graph theoretic approach 99

5.4 Symmetric closure 127

5.5 The algorithm 133

5.6 Validation 139

5.7 Summary and discussion 156

6 Graph Theoretic Operators and Morphisms in CCS **158**

6.1 Introduction 158

6.2 Rationale and definition 159

6.3 Application 166

6.4 Summary 169

7 Conclusions **171**

7.1 Summary of thesis 171

7.2 Results 172

7.3 Conclusions 176

8 Future research **179**

8.1 Introduction 179

8.2 Practical 180

8.3 Theoretical 182

Bibliography **186**

A Notation 211

B Example from Chapter 5 213

C Case Study 215

D Observational equivalence 243

E Iterative solution methods 245

F Quotient machines and generalised automata theory 247

List of Figures

1.1 Navigation chart for the thesis 3

4.1 In this example $\mathcal{M}_q$ does an implicit θ transition from q to q'. $\mathcal{M}_p$ cannot carry out this transition, hence it must be done by $\mathcal{M}_r$. Here we see that a series of transitions, both within and between K-sets is needed to find an equivalent transition in $(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A$. In this way $(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \approx \mathcal{M}_q$ if this can be done for all transitions in $\mathcal{M}_q$. In other words, that the K-sets are both I-complete and O-complete. . . 73

4.2 The graphs of $\mathcal{M}_r$, $\mathcal{M}_p$ and $\mathcal{M}_q$ 89

5.1 Illustration of the quotient graph mechanism. 97

5.2 Intrinsic and communicating τ-actions. The intrinsic actions, τ_i, are τ actions performed by $\mathcal{M}_r$ or $\mathcal{M}_p$ without communication. The τ_c actions refer to actions which have arisen as a result of $\mathcal{M}_p$ and $\mathcal{M}_r$ communicating. This graph shows the possibilities that may emerge from a structural perspective. Intrinsic actions will also possess a degree of symmetry (due to the properties of '|' and the GCP. Communicating actions will typically be assymetric, but can be symmetric (see the transitions involving $(p_1, r_3), (p_2, r_3), (p_3, r_3)$). The distinction between symmetric τ_i and symmetric τ_c actions is the fact that the former involves no actions from $\mathcal{M}_p$, whilst the latter does. Actions that $\mathcal{M}_p$ does, and which are involved in communication, will not appear in $\mathcal{M}_q$. 106

5.3 The Π and $\tilde{\Pi}$ planes 111

5.4 The $\tilde{\Pi}$ and L sets. The L-sets are the $\mathcal{M}_r$ equivalent of the $\tilde{\Pi}$-sets. Like the $\tilde{\Pi}$-sets, which are formed to deal with disconnected $\mathcal{M}_p$ machines, the L-sets are intended to address potential $\mathcal{M}_r$ disconnectedness. . . 116

5.5 Machine $\mathcal{M}_q$ 117

5.6 The projection of the split τ onto the $\mathcal{M}_p$ and $\mathcal{M}_r$ machines 119

5.7 $\mathcal{M}_q^\bullet$ – ready for quotient extraction 120

5.8 Possible τ derivations in $\mathcal{M}_q$ 121

5.9 Possible τ derivations in $\mathcal{M}_q$ with $(p_i, r_j) \rightarrow^\tau (p_k, r_j)$ 123

5.10 Possible τ derivations in $\mathcal{M}_q$ with $(p_i, r_j) \rightarrow^\tau (p_i, r_l)$ 125

5.11 Possible τ derivations in $\mathcal{M}_q$ with $(p_i, r_j) \rightarrow^\tau (p_i, r_j)$ 126

5.12 Symmetric closure example 128

5.13 $\mathcal{G}_{\mathcal{M}_p}$ and $\tilde{\Pi}$ 142

5.14 $p\tau$-graph 144

5.15 $p\tau$-graph with τ_i actions removed 145

5.16 $p\tau$-graph with split τ_c actions 149
5.17 $\mathcal{G}_{\mathcal{M}_\tau}$ 150

7.1 Processing capabilities of a selected range of algorithms 175

8.1 Relating topology and machines. 185
8.2 The underlying digraph structure of $\mathcal{M}_\tau$ 241
8.3 The mechanics of the iterative process. 246
8.4 The research in the context of generalised automata theory. 248

Acknowledgements

This thesis would not have been possible if it were not for the support from a number of individuals.

Firstly, my thanks go to Dr Mark Norris of BT for his initial enthusiasm in getting this project off the ground, and for supplying the necessary funds for the duration of the research - no mean feat in industry. I would also like to thank my employers BT, as represented by Dr. S. Stockman, Dr E. Cusak, Mr R. Lewis and Prof. P. Cochrane, for the patience extended to me during my studies. Thanks also to Dr R. Everett, Mr C. Barry and Mr C. Osborne for their assistance, especially in the early stages. I am indebted to Dr G. Martin for all the interesting discussions and debates, and for checking the mathematical proofs contained within this thesis. The quality and rigour of some of my arguments may have been less convincing without his input.

I would like to thank my examiners Professor J. Monk and Professor D. Andrews for there helpful suggestions. They have added greatly to the understandability of this work.

Before I embarked on this research I heard numerous warnings about working with the right supervisor. Well, I was one of the lucky ones. For four years Professor Ince has been guiding, nudging, provoking, stimulating and most importantly encouraging a student that was apt at going off at tangents. There is no doubt in my mind that without his help I would not have got this far, and I thank him whole-heartedly.

Last, but most definitely not least, I'd like to thank my wife for putting up with me over the past four years, and for checking my grammar. Hopefully the eurekas (usually false alarms) in the middle of the night are over now!

0.1 Publications

The following refereed papers have been accepted for publication.

1. A. Pengelly, D. Ince. *Protocol Synthesis using Graph Theory.* Proceedings Applications of Combinatorial Mathematics. Oxford University Press, 1995.

2. A. Pengelly, D. Ince. *The Solution to the Interface Equation.* International Congress on Applied Mathematics. 1995.

3. A. Pengelly, D. Ince. *Quotient Machines and the Interface Equation.* Computer Journal — subject to alterations.

0.2 Abstract

The synthesis and subsequent generation of protocol converters can be a time consuming and tedious affair. An automatic means of generating these converters, given a formal description of the interfacing protocols, has been researched by a number of academics and industrialists. From industry's point of view, the potential gains both in time and resource savings are large. While there is a strong standards movement within the protocol community, the output from that process is likely to be slow in terms of its dissemination to the communications world at large. The recent emergence of the Superhighway has compounded the problems of standardisation. Hence, the development of convertors will be a key issue for many years to come.

By far the most concise and complete attack on the problem to date came from a body of work referred to as the interface equation. While more general than other methods it has suffered from excessive computational complexity and, as with all other published work, a lack of demonstrable scalability. Also, the necessary and sufficient conditions before a solution could be found — if it existed at all — limited the domain of applicability.

The work in this thesis can be seen as a natural extension of this work, in that the problems of computability, scalability and a number of other issues, have been addressed, albeit by a different approach.

The theory developed arose from a completely novel approach — moving the problem into the domain of graph theory and topology. The resulting theory has spawned a number of interesting results, which include extensions to CCS, further development of the quotient machine concept, the notion of symmetric validation and a further unification between automata and graph theory. Indeed, it is expected that the merging of these two disciplines may well open a rich field of study.

The theory has been validated using numerous examples, including a complex 600

state synthesis problem which, while still falling short of real industrial problems, is at least 1.5 orders of magnitude larger than those published by other researchers.

Chapter 1

Introduction

This thesis reports on research undertaken on the problem of automatic protocol converter synthesis using formal methods and graph theory. The research is based on what the author considers to be the most developed thread of previous research — namely the CCS Interface Equation. The use of graph theory in this context is entirely novel and not only leads to a workable algorithm, but also raises a number of interesting results and deep research issues. The resulting theory enables a significantly wider class of protocol converter problems to be solved than was previously possible. This has been achieved by overcoming a number of key limitations in previous work.

It should be noted that, unlike many previous techniques, the theory is not limited to protocol synthesis, but can in fact be used to study any interface problem that can be specified in a meaningful way.

A key weakness with other algorithms is demonstrable scalability. While acknowledging that this is a difficult issue, a relatively large scale problem of 600 states was successfully solved. The graph theoretic approach has also elucidated much of the underlying solution mechanics of the interface equation, and has meant that previous

solution attempts would benefit from being revisited in the light of the results put forward in this thesis.

The structure of the thesis is shown in figure 1.1. Following on from this chapter, Chapter 2 introduces the general domain of formal methods — why they were developed and the problems associated with transferring them to industry. Developing a technique which could be usefully down-streamed was a major theme of the research. This chapter provides the reader with a broad overview of the range of formal methods from semi-formal notations such as SDL to temporal languages such as timed Petri-Nets.

The issue of application is a critical one for the formal methods community. Application areas are many but the one that seemed most interesting — and not only from the point of view of the authors industrial affiliation but also from the sheer number of technical challenges — was telecommunications and in particular protocols. As such, a comprehensive investigation of previous research on formal methods and protocol engineering was undertaken. The results of this survey are detailed in chapter 3.

The work undertaken by previous researchers fell into three categories — verification, validation and to a lesser extent conversion. Protocol verification and validation have been rich areas of study over the past 15 years, with a wide range of theories and techniques being developed. However, the practicality of many of these techniques was, and is still, open to question. Of more particular interest, especially in terms of the potential industrial gains, was protocol conversion. The synthesis and generation of protocol converters is a time-consuming and at times laborious process. The opportunities associated with developing automatic techniques were clear : reduced development cost and quicker time to market for the services in question. It was noted by the author that although a few automated procedures had been published, they all

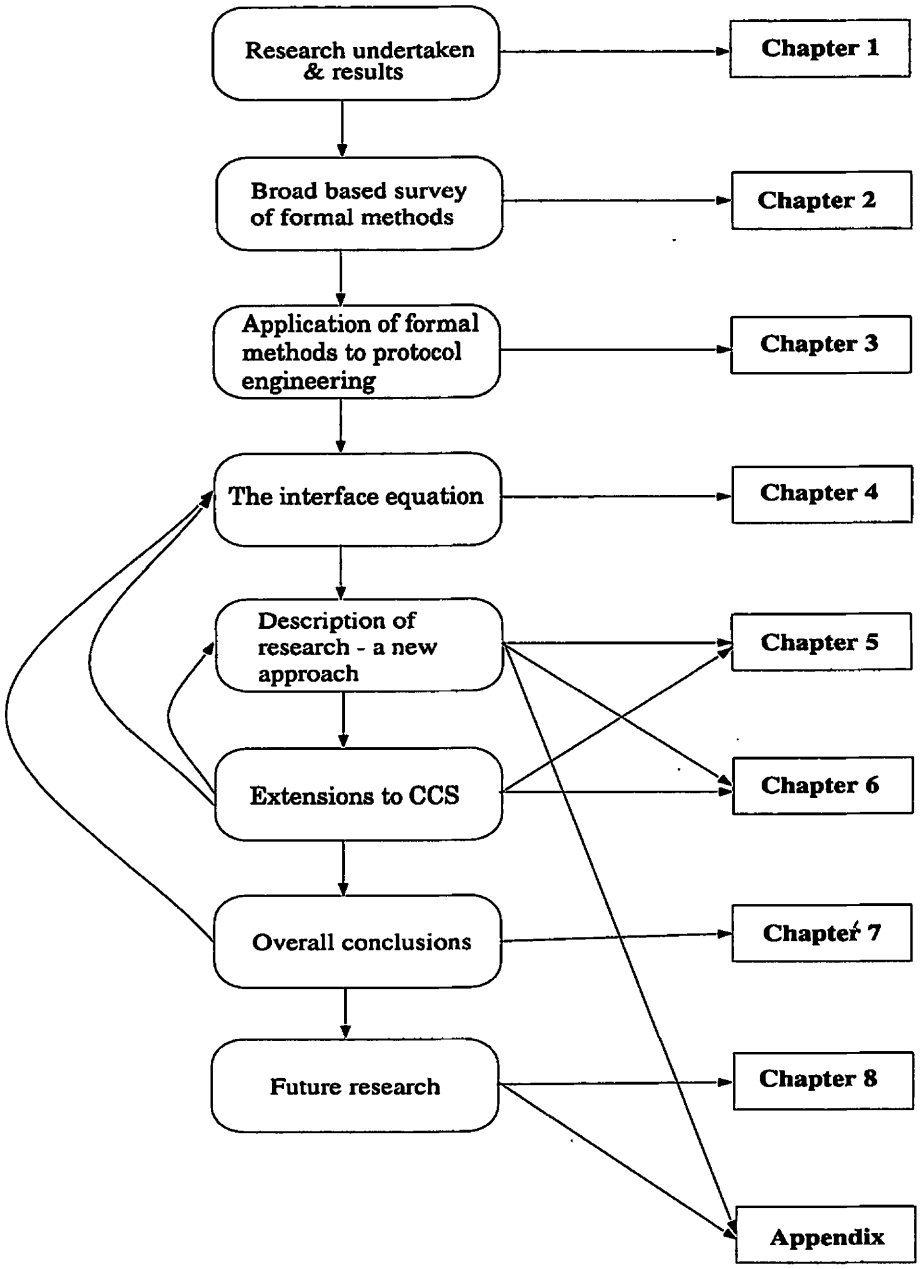


Figure 1.1: Navigation chart for the thesis

suffered from a number of drawbacks. Firstly, they left open the issue of scalability with, at best, unsubstantiated claims of application to large-scale examples. Secondly y, the most general algorithms for solving protocol converter problems were NP [94], with some authors claiming that any improvement of the computational complexity an impossibility. Thirdly, the techniques seemed rather brute-force in nature.

It appeared to the author that the general problem seemed to suggest the possibility of a more elegant and efficient approach and, more subtly, that the techniques (with the exception of one — the interface equation) were restricted to the protocol domain. This is fine given the overall context, but it occurred to the author that any protocol conversion technique ought to generalise to a more general theory of interface synthesis. It was in the light of these observations that the problem of automatic protocol synthesis was chosen as the topic for further research. Due to its potential generality, and the fact that it has a sizable body of results already documented [1], the interface equation was chosen as the basis for the research undertaken. An investigation into the interface equation was undertaken and is presented in chapter 4.

In chapter 4 the interface equation and previous attempts at solving it are presented. The chapter gives an overview of the work of Shields [217, 218] and Martin [142]. Shields discarding algorithm was the first attempt to solve such equations in a general setting. Whilst crude, the algorithm was successful. Martin modified the discarding algorithm and produced the constructive algorithm, which given the right initial conditions, was a significant improvement on the discarding algorithm. The chapter presents the main definitions and results, along with a small example of the application of the discarding algorithm. There are three reasons for presenting the work of Shields and Martin. The first is to introduce the reader to the body of work upon which this thesis is based. The second was to identify the weaknesses with

[1] It should be noted that many of these results were never published, but were available to the author.

previous solution attempts, and how they might be addressed. Lastly, the description of the interface equation was greatly facilitated by the copious use of graphs, and it was via the process of attempting to describe the work in a structural vein that the key idea of this thesis arose — to use graph theory to solve the interface equation. The basic idea was to apply quotient graph theory to what is essentially a quotient machine problem.

The use of graph theory required a different approach to the problem. The main properties of interest using this approach is the topological characteristics of the machines concerned. Symmetry is a recurrent theme, which spawned a number of interesting results, including the notion of symmetric validation and an extension to CCS called CCSg. Symmetry also provides the lever in terms of dealing with communication, and in extracting the quotient machine using quotient graph algorithms. The application of graph theoretic operators and morphisms is presented in chapter 5. In this chapter all the underlying theory and associated definitions are presented, along with examples to demonstrate the mechanics and validity of the algorithm. As a consequence of this structural approach, a number of proposed additions to the syntax of CCS were derived, and are formally presented in chapter 6.

The theory is validated via small examples in chapter 5, and a relatively large scale problem in the appendix C. While still not industrial strength, it represents a major increase of 1.5 orders of magnitude of scale on previous published techniques.

A critical appraisal of the work is given. This summarises the success of the research and where the results fell short of expectation. The application of graph theory to the problem of interface synthesis has opened up a wide range of interesting questions for further research. These, along with some possible approaches, are discussed in chapter 8.

Chapter 2

Formal Methods and Telecommunications

This chapter provides an overview of formal methods in terms of their diversity and type. It discusses the motivation behind their development and why research continues - particularly with respect to wide spectrum languages.

2.1 Software Engineering and Formality

In comparison to many other engineering disciplines, software engineering is still relatively immature [163]. Software systems are still produced in an *ad hoc* fashion. The quality of the resultant software is also suspect. Examples include metrics, requirements capture, process modelling and formal testing. One area in particular has received much attention, and that is formal methods [90, 113, 47, 100].

This is not to say that there is no formal basis for software engineering. On the contrary, there is a branch of mathematics which is ideally suited to computers. This is discrete mathematics [53] and it performs the same role in computer science

and software engineering as continuous mathematics does for classical engineering. The formal analysis of computers spans several decades [22]. But it was Boole with logic [18], Turing with abstract machines [57] and von Neumann with stored program computers [170] who really began to lay the foundations upon which modern computers are based. Indeed modern computer architectures are based on von Neumann's work. So there is in fact a vast array of results and theories. The problem is that the software engineering industry has yet to exploit this knowledge to any great extent, even for applications which would seem to be ideally suited, such as telecommunications system development. We shall briefly examine why this is and why application is important.

With traditional engineering approaches there is a wide body of knowledge relating to the *application* of the underlying formal theories. This inter-play between theory and practice is essential. Applying a theory (which is effectively a model) to the real world is a validation process. When applying a theory to the real world we hope the underlying model is a good one — that the resultant data or behaviour is an accurate reflection of the real process. If it is not, then the theory is modified to take into account new findings. Examples include cosmologists studying the large-scale structure of the universe, space engineers calculating the trajectory of an interplanetary spacecraft, and meteorologists forecasting the weather. In each case established theories underpin the practical work undertaken — and are modified on the basis of the theoretical model output.

The problem with software engineering is that the underlying theorems and models have not been embraced by the software community. There is a gap between the theorists and practitioners. The software engineer views the concepts of discrete maths and computer science as distant and esoteric — even though lots of academic research has been devoted to formal methods research. The idea behind formal methods is

to provide a medium whereby a system can be rigorously defined using an abstract language and subsequently checked using mathematical proof techniques [97] such as predicate logic. Thus systems can be developed which, in theory, have a drastically reduced susceptibility to errors of interpretation and content. This is particularly important for those systems that are to operate under severe performance constraints, for example those with high reliability, security and integrity requirements.

Whether it be the A320 Airbus or telecommunications network management software, the cost of failure can be very large. An A320 Airbus has already been lost due to inappropriate software [239] and in the UK an incorrect dating algorithm meant that 5 million customers were sent incorrect credit card bills [109]. Such events should provide a sufficient incentive to the software industry ensure that software is fit for purpose and correct.

The need to get such systems 'right first time' is a daunting task in many cases, and one in which traditional means of software development are largely inadequate. Techniques such as diverse development [225], system redundancy [225] and fault tolerance [184] have been deployed to assure safe software. But the underlying assumption with these techniques is that errors *will* exist and that the aim is to minimise the impact of such defects on the system as a whole. Hence, they address the symptoms, but not the cause.

That formal methods do provide an appropriate medium for systems analysis is not really in question. The problem is the gap between the level of abstraction and the problem itself is large. As a result the average software engineer, with perhaps little training in discrete mathematics or computer science, views formal methods with suspicion. And like traditional engineers trying to use tensors without an understanding of the transformation laws, software engineers find themselves with the raw materials, but not the tools to fashion it. In this situation the software engineer will

tend to use the process of trial and error for validation. From an engineering point of view there is nothing unethical about this. It is precisely the way engineers used to build bridges, dams and other structures before the relevant applicable mathematics was at hand! However, what it does mean is that it relegates software engineering to a *craft* industry [137, 108].

Addressing this gap between theory and practice is a key problem [1]. The current orthodoxy is that the development of appropriate tools is perhaps the best approach. A number of tools have and are being developed in an attempt to make the use of formal methods more palatable [49]. Inherent within these tools should be the basic theorems and techniques that correspond to the traditional engineers' chain rule, grad, div and curl and so on. The relative immaturity of these tools indicates that there is still much to be done in this area, and that the necessary facilitators are few and far between — though progress is certainly being made.

This thesis is concerned with one such 'facilitator' — the Interface Equation [165]. What it is, and how it works, will be discussed in later chapters. However, the principle is that of generating a formal specification of a system with minimal human interaction, thereby assuring its correctness.

2.2 Outline of Formal Methods

System specifications need to be written using a language. Up until relatively recently natural language was the predominant medium used. The expressive power of natural languages would seem to suggest that this is good practice; also no special training is required. But there are major problems using, for instance, English (a particularly expressive language) in a formal context [51]. Natural languages can be notoriously

[1] It should be noted that this problem is not limited to formal methods, but is a recurrent problem throughout virtually all aspects of enabling technology for software engineering.

ambiguous and are not amenable to formal, mathematical analysis (formal language theory is concerned with the grammatical structure of the language — not the subject it is describing [190]). This is due, in part, to the redundancy inherent within most languages, a characteristic which is actually invaluable for communication purposes. In short, there is no way of checking whether the system being described is complete and consistent in a formal and unambiguous manner.

In order to address these problems a number of alternative approaches have been developed. These include the use of stylized diagrams such as flowcharts and circuit schematics [5]. They can undoubtably be much easier to understand than natural language, but rely heavily on standard definitions in order to enable a consistent interpretation. Programming languages [34] can also be used since they are certainly unambiguous (assuming a compiler or interpreter exists) and open to analysis. However, they are generally too prescriptive and detailed, as they reflect a particular implementation.

Given the above there would appear to be a clear need for a class of languages which is more appropriate for system specification. Recent developments in discrete mathematics by Milner [152], Hoare [99] etc. have resulted in the emergence of a number of languages optimised for system specification [152, 99]. The mathematical basis for these formal methods (as they have become collectively known) means that they are unambiguous, open to very formal analysis and that they represent the system at a number of abstract levels to the point where there is very little, if any, design content. The disadvantage is that a degree of mathematical skill is required to use them. The problem of understandability is due in part to the *lack* of redundancy within these languages. This demonstrates a classical dichotomy - readability or rigour [51]?

Three general approaches have emerged, these are model (or state)-based, alge-

braic (or axiomatic) and process-algebraic specification techniques.

The model-based approach is characterised by being based on set and predicate logic. It is based on the general idea of a finite state machine. Such machines consist of states and actions (state transitions). In terms of analysis the machine has two 'modes', namely static and dynamic. The static mode defines the nature of the states themselves, whilst the dynamic mode defines the state transitions. So the languages define the legitimate states the system can be in and describe how operations alter the state of the system [35]. Examples of this approach are the specification languages Z [226] and VDM [116]. Such languages are most effective when specifying intensive data oriented problems, but less so for event-driven systems.

Algebraic languages are somewhat different and are based on the concept of abstract data types (ADT) [88]. In these languages collections of objects are described in terms of a set of equations or axioms (hence the alternative name 'axiomatic'), but without reference to concepts such as state or storage. Objects of a given type are produced by sequencing *constructor* operations which must be of the same type. Other operations of the type, called *inquiry* operations, produce different types and are the only means of extracting information from an object of the type. Perhaps the best known example of an algebraic languages are OBJ [81] and ACT-ONE [60]. A particular benefit of OBJ is that specifications may be executed and thus exercised. Hence, they are particularly useful for verification purposes in a range of applications.

The third approach is based on process algebras. Examples are CCS [152] and CSP [99]. These languages are well-suited for describing the behaviour of event-driven concurrent systems. More will be said on formal methods and concurrency in section 2.3.

Given the diverse range of application domains no single formal language can cater for all of them — although such a *wide-spectrum* language is much sort after. Such

a language would be used to specify all aspects of a system — including communication, data and interfaces. As things stand, each component uses a particular formal language. For instance, data-oriented systems use Z or VDM, whilst communication is best dealt with using CCS and CSP etc. Each has its own particular set of merits and pitfalls. Hybrid languages such as LOTOS [110] have been developed using ideas from the algebraic and process-algebraic approaches. ACT ONE [61] provides LOTOS with a data typing facility, whilst CSP and CCS deal with the behavioural aspects. As a consequence, LOTOS can be used for a variety of applications, but whether it is a truly wide-spectrum language or not is still an open question.

2.3 Formal Methods and Concurrency

The use of formal methods for data-oriented systems is reasonably well understood and the application of notations such as **Z** and VDM presents few conceptual difficulties. In contrast, the use of formal methods for highly concurrent, nondeterministic systems is more problematic.

2.3.1 Concurrency

There has been a growth of interest in concurrency on the part of computer scientists [148, 128, 193, 147, 214, 174, 215] during the last decade. A number of techniques and theorems have emerged on, for example, concurrent algorithms, programming languages and systems organisation. However, there is more to concurrency than making, say, Fourier transforms run faster. This is reflected in the breadth of application of some contemporary theories in which the computer is seen as belonging to a class of more general systems [3, 4].

But what is a concurrent system? A concurrent system consists of entities which can interact with each other in parallel. Each action of an entity is either an interac-

tion with other entities in its immediate neighbourhood (including itself), or it occurs independently of them. The first case describes the communication, whilst the second describes concurrency, namely different entities doing different things in parallel. A good example is the telecommunications network. To complicate the picture an entity can consist of sub-components (sub-entities), where these independent actions may be communications between these sub-components. This could be taken to extremes, but it does illustrate the potential complexity of concurrent systems.

Generalising somewhat, any system that may be regarded as being in part composed of entities, interacting to achieve some task and in the presence of constraints of a synchronic nature, presents organisational problems similar to those encountered in a telecommunication system — a human organisation is an example. As such, they could very well be susceptible to the same analytical treatment. This is well recognised by some theoreticians such as Petri, who defines a system as '...an environment for the organisation of information flow' and of the computer as a '..generalised medium of communication'. The central concept in his General Net Theory of processes and systems is concurrency [179, 14].

2.3.2 The Problems with Concurrency

A key characteristic of concurrent systems is that activities and events cannot be given a total ordering as with sequential systems, but can only be described in terms of a partial ordering. This uncertainty over the exact ordering of events is referred to as *nondeterminism.* This creates certain problems regarding the development of such systems. For instance, testing nondeterministic systems can reveal the presence of intermittent or transient faults. These faults are difficult to characterise since the precise order of events leading up to the failure are not always known. Thus the system exhibits what Bustard et al refer to as *time-dependence* [30]. Clearly this is a

property that needs to be avoided.

Concurrency by its very nature entails the simultaneous processing of a number of sub-systems. In many cases these sub-systems need to share resources, such as the CPU or memory space for example. This sharing of resources leads to two problems :-

1. How can each sub-system have fair access to the resources concerned?

2. How does the system avoid or recover from deadlock?

Problem (1), as the statement suggests, is concerned with the issue of *fairness*. Fairness is not a single property, but rather a collection of different concepts. Francez [78] for example gives three general classes of fairness, they are :-

1. Unconditional fairness

2. Weak fairness

3. Strong fairness

Lehman et al [129] uses the term *justice* to describe the requirement that a continuously active process will eventually be scheduled.

Another problem with concurrent systems is *deadlock* [152, 99]. Deadlock occurs when a process, having used resources required by another process, refuses or fails to release them. Thus the other process is unable to proceed and hence waits indefinitely. A good illustration of how deadlock can occur is given in [26] where the Dining Philosophers problem is analysed.

Any model of concurrency needs to consider these issues. The next section describes how a number of models and corresponding languages have been developed to address concurrent systems.

2.3.3 Formal Methods for Concurrency

Milner [152, 154, 151] was the first person to apply the process-algebraic approach to concurrent (communicating) systems when he developed CCS. His work provided the catalyst for many other researchers developed their own languages, for example Hennessy [95] and Hoare [99]. Milner's approach was to consider the fundamental behaviour of concurrent system in terms of *indivisible interaction* [152]. The elementary components of Milner's model are *senders, receivers* and *mediums.* Coupled with these basic concepts are a set of *combinators* or composition operators. Another important concept was bisimularity (equivalence), which was based on earlier work by Park [173]. Hoare [99], Hennessy [95], Bergstra and Klop [12] and Bekic [10] have developed algebras or calculi which are quite similar to CCS. Bekic's work differs in a subtle way, being based on the idea of actions on agents as opposed to communication. The Communicating Sequential Processes (CSP) developed by Hoare has proved to be very influential. Although based on the same underlying principle of indivisible interaction, it differs from CCS in a number of ways and has a richer syntax.

How do such languages deal with the expression 'the agent $\mathcal{X}$ is at one time in state **1** and later in state **2**'? Clearly there is a notion of *time* here which is not explicit in the languages so far discussed. The temporal dynamics of the system need to be specified in some way. This is particularly important when analysing systems that exhibit nondeterminism. Developing a logic capable of handling inferences involving time was first made by Findlay [74]. However it was Prior [185] who developed the first *tense logic.* This is in contrast to the approach adopted by Russell and Quine [186] who used essentially first-order logic to address the problem. Researchers such as Findlay and Prior were primarily concerned with the philosophical aspects of temporal logic. It was Pnueli [183] and Burstall [28] who developed the idea further, basing their work on Floyd and Hoare's inductive assertion method [76, 98] and considered

its application to software engineering.

Temporal logic has proved to be a particularly useful tool regarding the abstract, as opposed to the empirical, verification of systems. The incorporation of temporal reasoning techniques within existing specification languages has been studied by a number of researchers. For example Clarke et al [44] have used temporal logic for the automatic verification of finite state systems and Milner has developed a synchronous version of CCS [153]. An excellent overview of temporal logics can be found in [80].

2.4 Applying Formal Methods to Telecommunications

Telecommunication systems are highly concurrent systems which are becoming increasingly complex and large [52]. This complexity stems in part from the fact that they are assembled from various sub-components manufactured by different companies in different locations. This diversity creates numerous interface problems. It also reflects the increasing number of advanced services offered to customers. Such services have to be reliable and fast. The realisation of these networks requires a medium which is both flexible and relatively cheap — namely software. Hence, the modern telecommunications network is becoming increasingly *soft* : more and more of its functionality and intelligence is software-based [52]. Intelligent network management systems, automatic fault location, virtual networks and digital exchanges all require software in a potentially high risk (from a business perspective) role. Although there is an issue as to whether this software should lie in the network or at the customer premises, it will be there, and in large quantities.

Given this scenario one might expect telecommunications companies to be at, or near, the fore-front in terms of state of the art software development practice. But while a number of companies, such as AT&T, BT and NTT, have investigated advanced software technology few, if any, have embedded these techniques within

their development processes. This is not because of a shortage of techniques and methods — these exist in copius amounts. It stems primarily from the unwillingness of management to accept the extra cost at an early stage of the development process, without having the necessary faith that it will pay-off, in terms of added value, in the long run.

Another aspect of the problem is sheer complexity. Telecommunications networks and their communication protocols amount to a sophisticated system — which to specify using a formal language such as CCS or LOTOS becomes both a difficult and labour intensive task and significant overhead given current levels of knowledge and tool support. Yet current *ad hoc* methods can hardly claim to be more effective. The point is that they are better understood, that they sit more comfortably within what is essentially still a craft industry. There is little doubt that if the specification of a communication system already existed, then its subsequent development would in all probability be a significant improvement on these *ad hoc* methods. The problem, the author believes, is the specification process itself, which is itself a manual process and, as such, open to the same problems as code development. The automatic generation of specifications which are complete and concise is therefore highly desirable, but difficult to achieve in full [2]. The prospect of specification problems, which have been produced manually, developing in the rigorous

mathematical context of formal methods, which is completely beyond the everyday experience of the average software development team, [3] would clearly be a considerable worry for the software project manager.

[2] This thesis solves this problem to some extent.

[3] While many academic institutions now offer software engineering courses, the output in terms of trained engineers will not be available to industry for a number of years. In the meantime software development units will continue to be dominated by evacuees from other disciplines such as the arts and industrial sciences.

It is perhaps not surprising then that the focus of attention has swung away from specifying everything, to looking at particular subsets of the network. One such area is protocols; and these form the subject matter for the rest of this thesis. The importance of protocols arises from the fact that they form the nervous system of a network. They are responsible for ensuring that the various components act together as a harmonious whole [201] — particularly at the interfaces between inhomogeneous sub-components, where protocol converters are required.

The application of formal methods to protocol development has received a large amount of attention both in academia and industry. From the point of view of telecommunications application, is still the most likely target for any industry-based development.

2.5 Summary

The following key points emerge from this chapter :-

1. Telecommunications networks are becoming increasingly complex and are highly concurrent.

2. Software is becoming increasingly important to the telecommunications industry.

3. Current software development methods are largely inadequate and exhibit all the hallmarks of a craft industry.

4. Little of the wealth of advanced technology has permeated the (telecommunications) software industry.

5. The gap between theory and practice will close once a significant number of success stories emerge.

6. Automation and high quality tool support are basic prerequisites for successful technology transfer.

7. Protocols are a key component of the network and are an ideal vehicle for the application of formal methods and other advanced techniques.

The principle for those involved in technology transfer should be to start small, be successful, and then expand.

The next chapter explores in greater detail the application of formal methods to one such area — protocol engineering.

Chapter 3

Review of Formal Methods in Protocol Engineering

The world is richer than it is possible to express in any single language. –
Ilya Prigogine

Communications systems are some of the most complex that are currently built. They contain large amounts of hardware and software, are highly concurrent, and involve communication between components governed by formal rules known as protocols. While such systems are not as critical in terms of failure as say, safety-critical systems, they still exhibit a very high economic penalty for systems failure. The combination of complexity and high cost of failure make telecommunication systems good candidates for the application of formal methods. This chapter reviews the past application of formal methods to one aspect of telecommunication systems, namely their protocols. After a brief introduction to formal methods, separate sections examine their application to service specification, protocol specification, protocol analysis and protocol conversion, with each section concluding with a list of areas where research is still needed. The conclusion of the chapter is that, like many other promising application areas, formal methods for protocol engineering has given rise to much theory, but very little industrial application. The chapter closes by pointing out two areas, conformance testing and protocol conversion, where the application of formal methods could have significant commercial benefits.

3.1 Introduction

Chapter 2 provided an overview of the range of formal methods and languages available. The list is growing all the time, unfortunately very few of these methods have made the transition to industry in any major way. From the author's point of view the domain of application was important. As identified in chapter 2, telecommunications is considered an ideal area for the application of formal methods. Clearly, telecom-

munications consists of a variety of sub-disciplines, only some of which would provide a platform for any formal methods based research. Investigations revealed that the most appropriate area for study would be the protocol field. The main criteria was that of all sub-disciplines, this was the one that had the richest history in terms of formal analysis. This chapter presents the findings of an extensive literature search on the subject of formal methods and protocol engineering. The primary objective of this study was to identify a suitable topic for research that presented both a technical challenge, and which could in principle provide clear commercial advantages.

3.2 Communication Systems Development

Communications systems are highly complex. The early hardware-dominated telecommunication systems were complex enough; however, the introduction of stored program control systems, although greatly improving flexibility, has resulted in much greater increase in complexity. A further push towards complexity has been the move to integrated networks and services such as ISDN, IN and LANs [42, 176].

Communications systems are large mixed hardware/software systems which support very large numbers of processes where very reliable communication has to occur, not only between processes within a system, but between a system and other systems. There is a major issue of software correctness and reliability: since the cost–both direct and indirect–of a system failure, can be very large [52].

The combination of complexity and the high cost of failure has motivated the interest of communications systems suppliers and operators in specification techniques that can ensure the efficient production of accurate specifications and has often made telecommunication system suppliers pioneers in this area.

A major issue that permeates communication system development concerns protocols and the management of interfaces–not only within a system but also between a

system and another system. A protocol is a set of rules which allows communication between different system components that often have widely differing properties, for example, I/O port characteristics. Although the trend to partition a communication system into aspects which describe its logical, physical and conceptual behaviour–for example, via the OSI model–has undoubtedly led to a better understanding of systems, we are still faced with major problems. The solution to these problems in the areas of protocol analysis, specification and validation using formal methods of software development is the subject of this review chapter.

3.3 Formal Methods

A formal method of software development relies on the use of discrete mathematics for the functional specification and design of a system. A formal method also uses proof for validation; for example, in order to check whether a design is a correct reflection of a functional specification. There are a wide variety of methods available ranging from those suited to non-concurrent systems such as Z [226] and VDM [116], to those better suited to concurrent systems such as those found in the telecommunication area, for example CCS [152] and CSP [99].

While formal methods are not widely used, languages such as LOTOS which have a close link to formal methods of software development have generated considerable interest within the telecommunications industry evidenced by the relatively large number of tools now available, for example, tools such as LITE, LOLA, COOPER, Topo, Copos, Verif and Smile, and in projects such as the Esprit II[1] Lotosphere project.

The use of formal or semi-formal techniques for communications applications is not

[1] Esprit II is an information technology research and development programme funded by the European Community.

new. System X[2], for example, was completely specified using semi-formal message sequence charts and data flow diagrams [166] in 1979. The early pedigree of this project makes it, in terms of formality, or at least semi-formality, an innovative project–an example where the nature of telecommunications systems has forced developers to be more innovative than the vast majority of IT suppliers. The notation used was quite similar in many respects to the specification language SDL which was emerging as a potential standard at the time and which is now the most widely used 'method' in the communications industry [37, 207]. SDL (Specification and Description Language), emerged as a result of the CCITT[3] responding to the task of developing a language based on users' experiences. The basic language was available as early as 1976. It was further refined in 1980, with the most current version being released in 1988. Between 1984 and 1988 the definition of the language was improved, resulting in the CCITT issuing it as a recommendation, Z.100, in 1988. SDL has a large amount of supporting literature [207, 67, 37, 196].

SDL is oriented to a static type of representation which does not immediately support important functions such as animation and formal reasoning. Although SDL-88 now contains facilities for concurrency it still does not offer the degree of precision that mathematical notations such as CCS offers and also does not support the type of analyses of system properties such as liveness which more formal notations have been designed to support.

While SDL provides a reasonably mature platform for the development of communications systems many problems encountered in practice require a greater degree of sophistication or generality. It is these problems and potential solutions which are discussed in the remainder of this chapter.

It is also worth noting that a more formal but related language, ESTELLE, [136],

[2]System X is a major telephone system originally developed by the British company Plessey Plc

[3]Comité Consultatif International de Télégraphique et Téléphonique.

has also been used in a communications environment [68] but there are very few industrial accounts of its use.

3.4 Protocol engineering

Far-reaching technological advances in communication networks have enabled the interconnection of heterogeneous systems in order to provide services such as data transfer and the sharing of distributed resources. A protocol defines a set of rules for communication among these systems. The importance of protocols to the whole field of communications has led to the establishment of a new sub-discipline of communications engineering: *protocol engineering* [66, 114].

Protocols can be extremely complex and, while to a certain degree, informal design techniques have been successful in this area, it has also become apparent that they have also yielded a disturbing number of protocols with undesirable behaviour.

West presents the best overview of the protocols area in the eighties, and speculates on progress in the nineties [245]. While West's paper provides a good overview, it is not particularly comprehensive in that it is a very broad review, is non-specific, and is now somewhat dated. Hence there is a need for a more up-to-date and comprehensive review of the impact of formal methods on protocol engineering–something this thesis aims to do.

West points out that the presence of three international standards for formal languages is regrettable but, in comparison with the number of programming languages, this is remarkably few. Also, he states that although formal methods have undoubtedly shown their value in the development of protocols, their use in industry is less than that expected ten years ago. He sees investment in training and education, rather than massive research advances, if this trend is to be reversed.

Scharwtz provides a good overview of the application of formal methods for pro-

tocol standards [210]. He describes a range of notations and techniques ranging from simple finite state machines to more advanced temporal logics. His findings are consistent with those of other researchers detailed later in this paper: that the adequate capture of liveness traits can only be carried out with temporal languages; while from a purely functional viewpoint basic notations such as finite state machines are adequate.

3.5 The Service Specification

The service concept is one of the most important architectural concepts of OSI. It has been introduced in order for systems to be considered from the user's point of view with all low level details removed. What remains is the abstract behaviour of the system as perceived by an external observer (the user).

The service specification describes the services that the protocol will provide. It is an architectural concept which has a direct affect on the methodology applied to the service and the protocol which is subsequently defined. The service specification concept has acquired an increasing level of recognition by protocol designers [238] since the protocol is the logical implementation of the service [192].

Gotzhein and Bochmann [192] explore the potential of formally deriving the specification of a protocol providing a given service using a synthesis approach. In this paper services are described by expressions, where service primitives are combined using a small set of operators that can be found in languages such as LOTOS, CCS and CSP. The protocol is then derived from this service specification in a completely automatic way via a constructive approach. While being an impressive piece of work in intellectual terms, there are a number of gaps in this work that need to be addressed in order to make their approach generally applicable. First, the specification language defined does not have a defined semantics. Second, it is not capable of dealing with

synchronisation and data. Lastly, it assumes a reliable medium.

In [71] Fekete describes a case study using a multicast protocol which employed IO state machines to formally model the service to be provided. The paper shows how to use such a technique to prove the correctness of the protocol and its implementation. However, a criticism of the work is that it involved rather a small protocol; much more work is needed to establish the technique as a practical proposition for the telecommunications industry.

There is also some history of Petri nets being used for service specification. The main work in this area is due to Bourget [21]. He starts from the viewpoint that the service specification and the protocol specification should be described using Petri-Nets. He defines a local isomorphism between the marking graphs of the protocol net and of the net obtained by merging the protocol and the service nets, in order to test if there are conceptual errors in the protocol. While this approach certainly has its merits, there are a number of issues which are not examined in [21]. Firstly, is verification via isomorphism sufficiently general? Also is the claim in the paper regarding complete anomaly detection justified? This last point is open to question, since the author himself identifies an anomaly which the method cannot deal with.

Numerical Petri Nets (NPNs) have also been used to characterise the Cambridge Fast Ring hardware at a high level of abstraction [15]. The NPN model describes the service provided to users (the M-Access service) and formalises it in order to remove ambiguities. It is then used as a basis for the development and verification of the protocols using the M-Access service.

Because of performance constraints some protocols have to be implemented using dedicated VSLI circuits. Ho *et al* [31] develop a Petri Net-based language called HNET ('H' for hardware). They describe how a FSM description of a protocol using Petri Nets can be converted to a hardware description language HNET which is

suitable for manipulation by a silicon compiler. The problem with both this work and [15] is that a strong case for the use of Petri net formalisms over any other is not made–certainly from the point of view of verification and validation.

Gotzhein, in a key paper, [85] has investigated the requirements for a mathematical formalism that is a suitable representation of the service. The conclusion reached by him is that the degree of expressiveness required goes beyond that of current, standardised formal languages; for example a formal description of services requires a notion of past, present and future. Hence, many standard techniques such as CSP are unsuitable and a temporal approach is necessary [85].

Gotzhein presents a more expressive language which is based on temporal logic and is an extension of first-order linear time [80]. The language is applied to specify the view of the service provider and to specify the view of the service user. With the relatively recent advent of temporal extensions to languages such as CCS (TCCS [155]) and CSP (TCSP [119]), it would be interesting to revisit Gotzhein's expressibility concerns in order to establish whether the expressibility of these languages has been extended sufficiently to answer Gotzhein's concern.

As can be discerned from this section of the review there is major research needed in this area:

- There is further work needed to examine the de-merits and merits of notation such as CCS and CSP for service specification. For example, there is a need for case studies using these notations from which such information can be extracted.

- If notations such as CSP and CCS are not adequate for service level specification should they be extended, perhaps via temporal logic or should new notations be developed? Gotzhein makes a persuasive case for the latter [85], stating that expressibility is a key requirement for service level specification - a property which he feels is lacking in formal notations such as CSP and CCS.

- There is a considerable amount of work needed which examines the link between the notation used for service specification and protocol specification, should they as Bourguet [21] suggests be the same? If they were the same, then would we lose some important facilities necessary for either service or protocol specification?

- There is also work needed which is predicated on assuming that different notations will be used for service specification and protocol specification. Such a study would also need to consider itermediate representations between such notations in order to facilitate refinement.

3.6 Protocol Specification

In defining a protocol it is important to describe the context in which it is to operate and the services it is to support. The context is provided by the overall architectural (such as OSI), and by the particular layer in which the protocol will reside. The architecture will provide the protocol engineer with a set of constraints in terms of other interfacing protocols and routing strategy. The protocols behaviour - in terms of the services it is to provide - is determined by the service specification. This will describe what data is to be transported, security requirements, and provide a general functional specification. Hence, whilst the architecture can be seen as determining the external aspects of a protocol (interfaces etc.), it is the service specification which determines its internal structure. As a result, a protocol specification should include:

- A general description of the purpose of a layer and the services it provides.

- An exact specification of the service to be provided.

- An exact specification of the service provided by the layer below, which is required for the correct and efficient operation of the protocol.

- The internal structure of the layer in terms of entities and relations.

- A description of the protocol(s) used between the entities, including an overall informal description of the operation of the entities, a list of the types and formats of messages exchanged between the entities and rules governing the reaction of each entity to user commands, messages from other entities and other internal events.

The use of formal methods in protocol specification is a rich area with a healthy diversity of approaches. Bochmann and Sunshine [17], in a much cited paper, survey the notations and techniques available at the beginning of the 1980s. They examined state transition models, program verification, symbolic execution and the use of design rules. Holzmann, in a more recent work [103], has examined the current state of the art in protocol specification. He asserts the need for adequate specification languages to formalise the requirements and behaviour of the protocol and the need for effective validation techniques. He also states that important improvements in the work of the 1970s and 1980s have been made, most notably in behavioural specification, requirements specification and validation.

Venkatraman and Piatowski [236] classify specification techniques by defining a detailed set of characterising attributes. The following specification languages were investigated by them: State Architecture Notation (SAN) [181], Estelle [111], Petri Nets [179], Reliability Software Production Language (RSPL) [211], Production Grammars [92] and Temporal Logic [210]. The authors came to a number of conclusions; the key ones are:

- Petri nets and temporal logics should be used for analysis only and not specification.

- Layered specification techniques are preferable to monolithic techniques.

- SAN exhibited the most features, with Estelle a close second.

A protocol specification technique that relates concurrent system behaviour to agent activity and interaction at a high level of abstraction is discussed in [11]. The work described there exploits a concurrent programming technique based on a synthesis of two paradigms, namely agent-oriented and definition-based programming languages.

3.6.1 General notations for protocol specification

There has been a history of using general purpose notations suited for other applications for the specification of protocols. CSP [99] has been proposed as a formal description technique for OSI [227] mainly on the basis of facilitating verification.

The use of high level programming languages to describe and analyse communication protocols has also been investigated. Concurrent Euclid has been investigated in [162] by Nakakawaji *et al.* The authors claim that Concurrent Euclid is superior in understandability, modularity and maintainability, but do not make it clear as to what they were comparing it with. They also claim that development times were significantly reduced, but provide no data as to how much. However, as Concurrent Euclid has strict data typing and simulation facilities it is perhaps surprising that this language has not been explored to any great extent by other researchers.

Ada facilities such as timers and tasks provide the impetus for the work described in [34]. In [101] Hoffman considers a formal approach to the hardware implementation of protocols. Programming languages have also been used. In [197] Roisin, whose work is predominantly based on CSP, addresses protocol architectures and the design of entities and interactions with a view to verification via an assertion method. The descriptions are tested using OCCAM, a programming language based on the specification notation CSP.

In [232] Till and Potter used the formal specification notation Z to specify gateway

functions for a communications network. The authors raise the problem of the applicability of Z to the specification of concurrent systems; in particular, the problem of temporal overlapping of events, which may lead to inconsistencies in the system state. This problem is not satisfactorily dealt with in that work, though research by Fergus and Ince [73] provides a starting point for a more concise approach.

3.6.2 Specialised notations for protocol specification

By 'specialised' we mean a language which has been developed to address specific problem areas and encompasses hybrids. Knapskog [120] investigated three possible formal description notations for protocol specification and design for secure systems. The first was Abstract Syntax Notation no. 1–ASN.1 which is defined in a joint CCITT/ISO standard [38]. The second was SDL [37] and the third was Numerical Petri Nets (NPN) [248]. Based on this study Knapskog concluded that no existing formal methods were fully adequate for the design of secure systems. This is related in some sense to Gotzhein's expressibility problem–the application domain (secure protocols) is simply too complex and broad to be addressed by a single language.

A more recent method is described by Toussaint [233] in which the formal method derived is based on the approach adopted by an intruder. The method is applied to the Kerberos and X.509 protocols. The technique seems to address the majority of Knapsog's concerns.

Numerical Petri nets and SDL are further discussed in [174] where the PROTEAN (PROTocol Emulation and ANalysis) system was used to analyse protocol specifications using NPNs and the MELBA tool to generate code from an SDL specification. State explosion problems were countered by using reduction techniques on the NPNs.

Although they can be somewhat complicated, Petri Nets have been shown to be an excellent tool when designing protocols; in particular for service validation [21].

Temporal extensions of finite state machines for protocol specification are considered in [35] where a model checker for temporal logic formulae is presented for which properties such as deadlock and fairness can be investigated.

Sequentially phased reasoning [229] is a principle by which operationally concurrently systems can be viewed as sequential from a logic point of view. Stomp [228] applies this idea to the derivation of a broadcast protocol as a design principle, though not to its full level of generality. The work suffers from the lack of a suitable calculus which is able to capture invariants, though further research is stated to be at hand. This is a useful piece of work although, unfortunately, the application of sequentially phased reasoning does involve a large number of distinct steps.

A common technique is to try and form hybrid languages from currently existing ones. An example can be found in [130]. Here, finite state machines, CSP, and Abstract Data Typing (ADT) are combined with the best features from each technique being used. The hybrid language is applied to the IEEE802.3 MAC protocol [130]. The author concluded that the hybrid language proposed is both effective and appropriate for the formal specification of communication protocols.

PASS (Parallel Activity Specification Scheme) is a language proposed by Fleischmann [75] and is based on extended finite state machines, though it is essentially graphical in its approach. Fleischmann applies it to both a protocol and its service description. The general findings are positive–the author claims that the use of PASS resulted in the speedy production of protocol specifications and a significant reduction in development times.

Haas [89] has developed a novel approach based on the idea that events and actions belonging to a protocol, can be viewed as the language of a grammar with the attributes of the syntactic variables being used to control the state transitions of the protocol machine.

A non-executable high level language has been developed by Hoffman [101] for protocol specification purposes called LDP[4]. The language is essentially a hybrid between CSP and BNF which is intended to permit a concise and unambiguous specification of protocol characteristics. An algorithm for protocol verification is presented, from which a descriptive model is derived that is related to the hardware implementation.

The need for protocol standards which are precise has resulted in formal specification languages becoming an important tool in this area. LOTOS, in particular, has found favour amongst many researchers [234, 19, 79]. It has also been used to specify protocols outside the OSI arena. Tilanus and Yang [231] describe their experiences of using LOTOS and the environment LOTTE, to specify the ISDN D-channel layer 3 protocol (D3 protocol). The fact that LOTOS does not have a timer construct could have been a problem to the authors since the D3 protocol is very timer rich. Fortunately, the vast majority of timers in this protocol are deadlock preventing timers (set when entering the next state and stopped immediately after the first interaction).

The use of formal specification 'structures' for automated protocol implementation from LOTOS to C is an idea presented in [62]. The idea here is that, provided that the specification is written in a 'structured' way (in the classic sense of the word), certain constructs in the LOTOS specification will map to related constructs in the C code. A weakness in this work is that it applies to the process aspects only and not to abstract data types. Unfortunately, the authors conclude that a universal mapping from LOTOS to C is unlikely to exist.

There are a number of points that we would wish to make about the use of formal notations for protocol specification:

- It must be said that the impact of formal methods on protocol standards is negligible with natural language and diagrams being used in the main.

[4]No expansion of the acronym LDP is given in [101].

- The pioneering work by Bochmann and Sunshine [17] in which they surveyed various formal notations with regard to their utility for protocol specification needs updating. A number of modern notation such as CSP, CCS and UNITY [39] have been developed or matured since the eighties and have been used to specify protocols, for example [172].

- There is an understandable trend of either using existing notations or hybrids of existing notations for protocol specification. It is questionable whether this trend addresses the problems of protocol specification. For example, Venkatraman and Piatowski [236] point out that some existing languages are more useful for analysing properties such as liveness rather than specification. Further work addressing this issue is clearly needed.

- There is an open question regarding the development of a special-purpose language for protocol engineering. Existing specialised languages have strengths and weaknesses. For example, the facilities for the specification of processes is excellent in LOTOS, while the data specification properties are limited. The first point for anyone carrying out research in this area is to ask what should be the core set of facilities for such a language.

- Protocol processing is carried out by hardware or software, with there being a distinct trend towards software. However, certainly in the medium term, we would envisage the role of hardware to be important. Consequently, a question that needs to be addressed is whether current notations are adequate for both media. Claims have been made, for example, that the UNITY notation [39] is not only target architecture independent, but also independent of a software or hardware implementation. Such claims need investigation and also the feasibility of current notations or extensions with regard to hardware and software need to be addressed.

- Protocol specification refinement techniques need to be investigated. Very little research seems to have been carried out in this area.

3.7 Protocol Analysis

The analysis of protocols centres on the behaviour of the communicating protocol entities and the underlying medium, i.e. the set of possible sequences of events that can occur when the entities execute. Protocol properties are often specified in terms of assertions associated with a transition system. Such properties are classified as either *safety* or *liveness* properties. A protocol is 'safe' if no undesirable event ever occurs, for example, deadlock. It is 'live' if all desired events eventually happen, for example, completion, recurrence or progress. Safety properties can be described as invariant assertions associated with transition system models, or as relations (simulations or bisimulations) between a proposed and a required behaviour. Liveness properties generally require temporal reasoning; hence the use of temporal logics becomes necessary. Temporal logic formulae which capture the liveness conditions are added to a specification and the state transition system checked to see if it is a model for these formulas.

The question of automated tool support for protocol analysis has been addressed by Holzmann in [102]. He points out that for a large proportion of protocols (less than 10^7 states), current tools such as Pandora, S/R validator and the Carnegie-Mellon model checker are practical for some problems. However, the increased size and complexity of modern protocols would seem to render them beyond the capabilities of what is currently available and, moreover, what is likely to exist for some time come.

3.7.1 Validation and verification

The issues of validation and verification have become important topics for study with regard to communication protocols. By *Validation* we mean the process of ensuring that a protocol meets the properties specified in the relevant section of the service specification. By *verification* we mean the checking of the output of a development phase against that of a previous phase.

The limitations and possibilities of automated protocol analysis is discussed in [102]. In principle it is possible to completely validate protocols, but in practice there are time and space constraints which prevent exhaustive analysis. This is perhaps the main problem why such tools are not used. The author introduces a simple symbolic execution method based on vector addition. This method is claimed to be two orders of magnitude faster than previous methods. Protocols of the order of 10^6 states are asserted to be analysed in minutes on a VAX750. Holzmann feels that protocols of this size are getting near to the practical limits of most algorithms and while there have no doubt been hardware improvements since the paper was written, no evidence has been found to suggest that larger protocols can be automatically validated in a reasonable time frame.

Validation

Lin *et al* [134] assert that to successfully validate protocols three ingredients are required: formal modelling; decomposition and abstraction; and reachability analysis. This is particularly true when attempting to validate very large protocols [134].

There is a diversity of techniques for validation; the developer is faced with a bewildering array of techniques, each tailored to highly specific problems, with very little else in common. For example, the models developed to address validation cannot, in general, be extended to include verification or testing. Another example is

that specification languages intended to capture the implicit complexity of protocols can sometimes be of little use to the tester, and so on. A general overview of the area is given by West [245] who points out that formal methods, as well as having a marked positive effect on the reliability and complexity of protocols, are likely to become crucial to the development of communications systems in the future.

The use of randomised techniques for verification purposes has been studied by a number of researchers and industrialists. The general approach is to identify the 'most likely' behavioural characteristics that a protocol will exhibit. This is certainly the approach of Maxemchuk and Sabnani [157]. Their method performs a partial exploration analysis of the protocol behaviour. Both they and other researchers (e.g. West [244]), conclude that random state exploration techniques can be as effective as exhaustive state exploration, and that random techniques are a significant addition to the protocol engineers tool kit. Systematic, or exhaustive state exploration suffers from two problems:

- the sequence of transitions executed is dictated by the depth-first or breadth-first ordering of the tree exploration, as opposed to the actual sequence of transitions.

- state explosion occurs, where the number of paths through the tree exceeds the capacity of the validation system.

The disadvantage with random methods is that it is difficult–if not impossible–to make precise statements about the correctness of a protocol, particularly as no well-defined notion of termination exists. However, similar limitations exist with more formal approaches, as described in [41], where CSP is used to analyse a protocol and obtain partial correctness. As protocols become more complex, it may well be the case that there will be definite limits on the degree of formality of validation that one can achieve; this suggests a key future role for random techniques combined with

formal methods.

Verification

Verification is a particularly pressing problem, and accordingly receives the most attention amongst researchers. Pehrson [176] provides a good tutorial on verification in the context of OSI. Emphasis is put on the methods and languages where algorithms are available and can be integrated into design tools. Equivalence and model checking via temporal logics is also addressed.

A different approach is to tailor the specification to suit the purposes of verification. This is done by Najm using LOTOS [161] who uses the term *Verification Oriented Specification* to describe this approach. The verification decision mechanism is based on comparing finite labelled transition systems using a tool called SCAN and on using observational equivalence.

Pachl [171] concentrates on the contents of the communication channels as a means of reasoning about system behaviour. A formalism, called *channel expressions*, is presented in relation to reachability analysis and is used to prove a decidability result about deadlock detection for communicating finite state machines (CFSMs). This work is based on a substantial research base on *recognisable* and *rational* relations. Pachl proves that deadlock freedom and the absence of unspecified receptions is decidable for protocols expressed via CFSMs, provided they have the recognisable channel property[5]. Unfortunately, these results do not hold for all CFSM protocols. The work described in this paper would benefit from further research in three areas. The first is to characterise the set of protocols for which the work is applicable. The second would be to examine other reachability properties. Lastly, whilst Pachl is unconvinced as to the feasibility, it would still seem valid to derive suitable algorithms and to automate

[5]This does not mean that there is an algorithm which will decide whether such a protocol is deadlock free or not.

these.

In [255] Yodaiken and Ramamritham apply a modal (state dependent) logic to the problem of investigating the fault-tolerance of a reliable net protocol. Yodaiken asserts that the language used (modal primitive recursive functions) has great expressive power, and is significantly more tractable than process algebras such as CCS and CSP. The authors claim that an extension of the language used can also help treat recovery, liveness and real-time performance, though unfortunately this extension is not documented or referenced. While the method clearly can be used for verifying certain aspects of safety, it's application to the general problem of verification is suspect and remains to be established.

A three way handshake connection establishment protocol is verified in [115] using temporal logic. The authors claim that specification of the protocol using temporal logic is sufficient to guarantee the total correctness (liveness and safety) of the protocol. The use of the conceptual decomposition of the protocol into entity and system level is stated to have been extremely useful, providing an ideal structure for the subsequent verification analysis. They also state that proving the safety properties first greatly facilitated the subsequent proof of the liveness properties. This isn't immediately obvious from [115] and warrants further study as to it's generality. However, the basic assertion that the use of temporal logics greatly enhances the analysis of protocols is consistent with other researchers results and further vindicates the claim that any prospective wide spectrum language must include a temporal dimension.

A similar approach is adopted by Shasha *et al* to investigate a carrier sense LAN protocol [213]. They compare classical temporal logic with interval temporal logic. They conclude that the interval temporal logic is more suited to the analysis of contention protocols than classical temporal logic, while for contention-free protocols the reverse is true.

Miller [150] provides a useful overview of the subject of verification and explores four verification formulations–CSFM, communicating machines with shared variables (CMSV), Petri-Nets and a simple programming language (SPL). In surveying the techniques available at the time, he attempts to encapsulate them into a unified whole. A number of gaps appear; in particular Miller argues that the need to specify systems in a clear and concise manner, using suitably expressive languages is at odds with the needs of formal verification, which is a simple, mathematically tractable model that lends itself to formal analysis. In essence Miller claims that there is a gap between specification and verification in terms of there being no single generic language which fulfills all the needs of both camps. However, it can be argued that some languages come close to filling this gap—for instance UNITY [39]. The basic requirement for an expressive language with basic operators would also seem to be addressed by CCS and CSP. But Miller does not consider process algebras or other notations such as Z and LOTOS.

The need for temporal extensions has been articulated by Sajkowski in [204], which describes a new approach to the application of time augmented reachability analysis for protocol verification purposes. Sajkowski found that the complexity of the creation of time expressions and the resulting state explosion leads to certain limitations:. Possible solutions to these problems are given in the form of rules for the derivation of the time expressions.

A theory of deductive systems and its application to protocol verification is examined in [132, 105, 241]. This theory is aimed at addressing the characteristics of protocol development which can be very complex: namely those associated with flow control and synchronisation. However, it is not clear what such an approach offers over the other techniques described in this section.

In summary:

- There are a bewildering array of techniques for protocol verification and validation. There is an urgent need for some form of rationalisation. Further research in this area should endeavour to identify the strengths and weaknesses of each technique.

- One area which is very advanced in terms of theory involves issues such as fairness and liveness, for example, [78]. However, much of this work has yet to percolate into the communications arena. This is almost certainly due to the inadequacies of the specification languages used in this application area and the fact that the specification of concurrent systems is still an embryonic area. Whatever the reason this area is one which holds a large number of challenging research questions.

- Lin et al [133] have pressed the point that to successfully validate large scale protocol systems one must decompose the system in some way. There are a number of techniques that are available for moderately complex systems, for example, Lam and Shanker's method of projections [126]. But these techniques require a high degree of intuition to actually construct a converter.

  Current communication systems are getting larger and more complex and there is now pressure to develop alternative, more practical techniques of decomposition. There is an issue of scalability with current techniques which needs resolving. Any new technique will need to demonstrate it's applicability to realistic examples.

3.7.2 Testing

By *testing* we mean that subset of activities which give rise to the execution of an interface with test data in order to check that a protocol meets the relevant section of the service specification.

The advantage of adopting a formal approach is that it presents the possibility, at least in principle, of automatically generating test suites from the relevant specifications. This assumption is based on the idea that it is possible to define mathematically provable transformations on a specification to produce a test suite.

There are two major aspects to protocol testing [222]:

- Static conformance testing–defining the allowed minimum capabilities of an implementation to facilitate inter-networking. The term static refers to the fact that it is the structure (defined by the associated program control flow for instance), which is the key property. Static tests do not require the protocol to be in operation. It merely requires the source code or some other (static) representation.

- Dynamic conformance testing–defining the set of allowable behaviours of an implementation in the instances of communication to check dynamic behaviour. Dynamic testing requires the protocol to be operational - these tests can only be carried out when the protocol is in use (usually by simulating usage in a controlled and deterministic manner).

An interesting method for protocol testing is described by Naik *et al* [159]. While they concentrate on LOTOS for specification and TTCN (Tree and Tabular Combined Notation) for test specification, it is clear that the method proposed will work for other languages such as Estelle and SDL. LOTOS and TTCN are not immediately compatible, hence some transformation was necessary (in this case on the LOTOS specification) in order to make the method viable. This was achieved by transforming the LOTOS code into EFSMs (Extended Finite State Machines). Indeed, it is this that makes the method independent of the specification language, since in principle any language can be transformed into an EFSM.

A promising approach to dealing with non-determinism in specifications is presented in [65]. The authors provide a theoretical foundation for a metric derived from a probabilistic model that is used to estimate the number of times that a test sequence should be tried in order to arrive at a fair verdict on the test results. They have developed an algorithm which has been implemented in a fully automatic test generator which accepts specifications expressed in textual LOTOS, graphical LOTOS or Petri Nets.

Because of the increasing complexity of modern protocols, generating tests by manual or *ad hoc* means is a task fraught with difficulty. Sidhu and Leung [223] propose four formal methods for generating test sequences for protocols, referred to as the T-, U-, D- and W- methods. The T-method is based on the work of Naito and Tsunoama [160] and involves the analysis of transition tours. The U-method involves the derivation of a unique input/output (UIO) sequence for each state and a β-sequence. It is based on the work described in [202], which presents a procedure for generating test sequences for checking conformity using finite state machines. The concept of a UIO (described as an input/output behaviour that is not exhibited by any other state) is introduced here. The D-method also centres on the construction of β- sequences, but every occurrence of a UIO sequence for a state is replaced with a distinguishing sequence [83]. Lastly the W-method involves the characterisation of a set, referred to as W, of the finite state machine. Details can be found in [43]. All the methods assume a minimal, strongly connected and completely specified Mealy machine [216].

Sun *et al* [230] present a number of new UIO sequence classes for conformance testing, verification and validation which aim to reduce the number of test sequences. They define the concept of adaptive UIO sequences which result in test sequence generation being based on certain characteristics of the UIO sequence rather than by

the use of concatenation.

Another method, the P-method, is described in [222]. The extraction of test scenarios from a LOTOS specification has also been the subject of research [146], where the OSI transport service model is analysed with respect to valid and invalid interactions.

Because of the complexity of recent communication protocols, generating conformance tests using manual or *ad hoc* techniques is virtually impossible [50]. Conformance testing is a very active research area at the present time. There are three distinct conformance testing methodology approaches: traditional methods [194], ISO 9646 (Conformance Testing Methodology and Framework) based methods [112] and formal methods-based techniques. Only the last approach will be discussed here.

Dahbura *et al* [50] present four formal methods for generating conformance test sequences for protocols. These are: transition tours, distinguishing sequences, characterising sequences and UIO sequences. Transition tours are best suited to the control of protocol tests. The distinguishing sequences and characterising sequences were particularly useful with respect to the observability aspects, whilst the UIO method was generally able to deal with both of these. They also list a number of interesting research issues. Among these are:

- Can performance testing be developed such that integration and interoperability testing is not necessary.
- What sound methodologies are needed to automatically generate useful high-level models of protocols for conformance testing purposes.

LOTOS [23, 234, 19] has received a great deal of attention with regard to conformance testing as it was initially developed to assist in the formal definition of protocols and services for computer networks; in particular those defined by the OSI Reference Model. The conformance relation proposed by Brinksma [25, 24] is the best

developed theory for LOTOS. Brinksma showed that each LOTOS specification has a unique canonical tester, which captures the test suite.

Pitt and Freestone [182] have examined the derivation of conformance tests from a LOTOS specification. The test processes constructed preserve the structure of the protocol specifications via a set of laws pertaining to the basic LOTOS operators. This theory is related to Brinksma's theoretical notion of the canonical tester.

Using Brinksma's work as a basis, Wezeman has developed an algorithm, the CO-OP method, for deriving canonical testers from their specifications [246, 247]. A canonical tester is a specification (in LOTOS) of the testers behaviour. It is an intermediary step in obtaining a test suite. In this context it is an encapsulation of all possible tests. The purpose is to ensure that a lower level LOTOS specification B conforms to a higher level LOTOS specification A. The canonical tester is produced from specification A. The actual testing, termed *test application*, consists of taking each test and running it in parallel with the specification B. The outcome of the test run is *Pass* if it is deadlock free. Given that LOTOS is currently used for the standardisation of OSI [19, 182, 48], this work presents a significant advance. We would regard this as a landmark in the application of formal methods to protocol testing. Recent work in the area of conformance relations has involved the development of event FSMs from LOTOS specifications [36]. These are then used as the basis of test generation using the UIO method.

Liu and Wang [135] describe a test suite generation method for conformance testing of protocols where the protocol specification models have memory. A verification technique, based on axiomatic semantics, is used to test protocols specified using the EFSM notation. The authors claim that by appropriate manipulation of the execution paths in the EFSM specification, the observed events can be used to examine the correctness of the protocol implementation, although little evidence is provided to

back this claim up. However, if this claim was true, then such a technique is capable of providing industrial-strength testing methods. Similar work using Estelle has been carried out by Favreau *et al* [68] who considered automatically generating partial test scenario skeletons from protocol specifications written in Estelle. The work is based on two principles:

- partitioning the abstract machine and considering only subsets,
- generating the test skeletons for each subset.

Conformance testing with respect to distributed systems addressed in the context of multiple observers (external agents interacting with a process) is described in [56]. Here the 'observer' is distinguished by being:

- Active–an observer can change specific control messages in order to control the test execution.
- Passive–the observer does not disturb the test system in any way.
- Local–the observer views one interaction point.
- Global–the observer views both interaction points

There are a number of important points to be made about this aspect of formal methods and validation:

- The work by Brinksma demonstrates how formal methods can indeed provide a useful addition to the protocol engineers tool kit.
- This has been a particularly active field of research where the clearest gains for formal notations are evident. Testing protocols is a time-consuming, error-prone activity and any notation that promises to at least partially automate the process becomes an economic proposition–even if that notation has a high degree of mathematical formalism.

- There are a number of good tool sets associated with notations such as CCS (the concurrency workbench), LOTOS (the LITE toolset) and SDL (GEODE, the SPIN toolset). However, some gaps remain. For example, little work has been carried out in developing tools which are capable of generating an optimal number of tests that maximise coverage of states, extended from formal languages such as Z and LOTOS.

- Whereas LOTOS [247] and Estelle [68] have been addressed with regard to tools for conformance testing we are unaware of tools which have been developed for other notations and other testing activities.

3.7.3 Performance analysis

Performance analysis is an important, but generally under-researched, subject. Assessment of protocol performance based on image protocols is described in [121]. The primary advantage of this approach is the reduced state space that is processed. Along with liveness and safety, it is shown that the image protocol is also faithful regarding performance, a result which simplifies protocol performance analysis considerably.

An elegant extension to Petri-Nets for the same purpose is described in [131]. Colour Generalised Stochastic Petri-nets (CGSPNs) are used to model the performance of integrated systems. CGSPNs are extended by introducing priority firings for different colours. The Markovian characteristics of the CGSPNs underlying stochastic processes is preserved by the extended model. The extended CGSPN is applied to an integrated data-voice system, for which protocol performance is evaluated.

The authors conclude that extended CGSPNs are suitable for integrated system, multi-layer, protocol performance evaluation in that they are able to model and analyse synchronisation, concurrency, co-operation, resource contention, class identification as well as priority services. However, a key problem which hinders application is

the prohibitively large underlying Markov chain. The above notations are mathematical, but are ideal for developing performance models. It remains an open question as to whether languages such as Z and LOTOS can be used in this context. With performance analysis increasing in terms of importance, there is a need to undertake research in this area.

3.8 Protocol Conversion

Even in these days of OSI standardisation there is an increased need for the development of software or hardware which acts as an interface between two possibly widely differing protocols. There are a number of reasons for this. Green [86], for example, gives a convincing set of reasons as to why OSI will never become a global standard. First, it is already too late, with very large, installed bases of SNA, DECnet and TCP/IP networks. Second, computer communication is a relatively young field and is improving all the time with new technology likely to destabilise any attempt to converge to a single architecture. This is particularly noticeable with respect to the Superhighway [84, 55, 87], where the issue of protocol standardisation is already proving to be a vexing problem, given the scope of the proposed network. It is the author's view that in all likelihood, no standard will be possible. And as such, if the Superhighway is to materialise, it will do so via the copius use of converters. Because of these reasons we feel protocol conversion will be an active research area–certainly in the medium term.

A number of researchers have been active in this field with [86, 221, 33, 124, 125, 169, 220] being the most cited work. The common feature of the work that has been carried out is the reliance on some form of formal description technique, usually simple finite state machines or communicating finite state machines. The general design principles employed are discussed in [16], as are the underlying models

in [254], which suggest strategies for solving the converter problem in a systematic manner. There are two basic approaches. The first derives converters at the service level by concatenating common services between the given protocols. In doing so, the protocol engineer can ignore the details of the PDU sequences and message exchange control. The problem with this is that since PDU level functionality is not always apparent at the service level, the reliability of the resulting protocol converter may be questionable. This is in fact the most common method (as will be seen below). The second is at the protocol data unit (PDU), where the service specification is ignored. The PDU approach has led to more formal techniques and where the influence of automata theory has been evident. The approach developed in this thesis, in common with a number of others, is of the latter kind.

Perhaps the most referenced work is by Green [86]. Green's approach begins by analysing the access path between two end users at the *discontinuity* point. This point is where a path passes from one system, obeying a certain architecture to the other system using a different architecture. This discontinuity can be arranged to occur at a number of nodes along the access path. Green's procedure starts by investigating the set of atomic protocol functions that support the services provided at the conversion point in terms of the interface between the two systems. By identifying a usable common subset, the nature of the mismatch can be identified. A hard mismatch means that one or more complementation layers and headers are added. If it is a soft mismatch then gateway code can be written to perform the conversion, possibly using formal specification; for example, by using finite state machines.

Cookson and Woodsford [145], describe an SDL-based methodology for the construction of protocol converters. The authors assert that a key characteristic of the methodology is that it stresses the need for good architecture, which in turn facilitates the control of the subsequent development process. Indeed, in some cases the

development effort has been halved. The method is heavily interactive-tool based (the authors do not specify the tools), particularly for design, verification and implementation, where tool-assisted code generation is used. The example given is certainly industrial strength (being a DASS2-DPNSS converter). However, it is our view that the use of SDL as the base language is somewhat limiting. Certainly there appears little scope for automation, a necessary step if development times are to be greatly reduced.

Chang and Liu [40] have transformed the protocol conversion problem into the protocol validation problem. The functions of a converter are expressed via the interaction of several processes which are represented by communicating finite state machines. The approach is based on Okumura's conversion seed concept [169], which specifies the semantic relationship between the messages of two given protocols. However, their work differs in specifying the conversion seed as a summary of restriction rules for protocol conversion. This amounts to generating several interacting processes, thereby allowing the use of established protocol validation techniques.

This overcomes a weakness in Okumura's approach. However, because it is restricted to the static properties of CFSMs, it limits the usefulness of the work. By relating the conversion problem to that of validation, the author's have been able to arrive at a method which is based on existing theory and which is quite general. It is asserted, for example, that with minor modifications to CSP the technique is viable for that notation. Unfortunately, no information is given as to the expected savings in development time and to the issue of automation. In addition, no statement is made as to the protocol classes to which the method can be applied. Needless to say, this looks a promising method and would no doubt benefit from further research, particularly on large complex protocols.

In [220] Shu and Liu describe a model to synthesise a converter by inserting

synchronisation points into two given protocols. Using CSFMs each synchronisation point consists of a pair of operations PUT and GET. When such a point is reached in one of the protocols the system idles until the other protocol reaches the same synchronisation point. Then the PUT and GET operations pass a message between the protocols. This work was later extended by defining the PUT and GET pair as asynchronous operations [221]. Also, whereas in [220] no buffer was employed, the latter work produces a converter with a non-FIFO buffer. The complexity of the solution is reduced by considering only the most significant messages between the two protocols. This approach simplifies the construction of the converter at the expense of expressibility, since it may not always be possible to represent all the constraints on the conversion process. The problem with this work is that the authors do not provide sufficient evidence that their method is superior to any other, in terms of dealing with complex conversion problems. Indeed, the examples used in the paper are quite trivial. In order to provide confidence in this work it really needs to be applied to protocols orders of magnitude higher than those so far used.

In [33] Calvert and Lam describe a number of interesting approaches to the problem of protocol conversion. They adopt the very pragmatic view that conversion is not an end in itself and that for only a certain class of mismatch problems will conversion be a reasonable solution. They describe three solutions: conversion by projection, Okumura's approach [169] and the quotient approach.

In [126] it is shown that protocol projection could be used to analyse the correctness of conversion methods and possibly derive a converter specification. The essential idea is to partition the protocol into a number of components such that the states within the component are indistinguishable in the image of that component. This induces an equivalence relation on the set of messages transmitted and received in the protocol. If two protocols P and Q can be projected onto the same image

protocol R, then R represents same common functionality between P and Q–it is a common image.

Unfortunately this approach is dependent on R having a sufficient functionality relative to the service specification for the converter and even then a heuristic approach is required to derive the converter itself. However, the approach does provide a sufficient condition for finding a converter. The projection preserves liveness and safety: that is any safety property that holds on the image protocol must hold on the original and any logical property (safety or liveness) holds for the image if and only if it holds in the original. The quotient approach is particularly interesting in that it offers, in theory, the most general method of solution. The concept of quotient machines has existed for sometime [216] and, in principle, it is possible to find the quotient of any pair of machines that can be expressed formally by, for example, using finite state machines.

Ohara *et al* [167] adopt an interesting approach to complexity reduction for protocol conversion. The inputs and outputs of the relevant finite state machine representations of the protocols are partitioned in three ways:

- Serial or service interface conversion–the finite state machine is decomposed into a sequence of finite state machines in series. The disadvantage of this approach is that it can be slow.

- Parallel or protocol interface conversion–the finite state machine is decomposed with respect to the various layers of the protocol. Although more efficient than the serial method, it does have the disadvantage that certain conditions need to be met if the conversion is to work.

- Hybrid conversion–this method adopts the best aspects of the preceding two. It is efficient and has a wide conversion coverage with few constraints.

This notion of complexity reduction is also pursued by Saha [203], who build on Okumara's conversion seed method [168]. Of particular interest in this paper are the rules for reduction prior to converter construction and the subsequent composition rules which are applied to the generate the full converter.

In [167] an application of the above to a DCNA Digital Corporation Network Architecture) and SNA (Systems Network Architecture) conversion problem is presented. The authors conclude that the work was effective in producing a compact conversion system, though a decomposition algorithm needs to be found.

Okumara has developed a formal methods based algorithm which requires aprior knowledge of the required services and the notion of a conversion seed [168]. The conversion seed is a finite state machine representation of the significant message relationships between the protocols. A converter is constructed by the algorithm from the Cartesian product of the state-space of the protocols using the seed behaviour as a template. The problem here is that deriving the seed is in itself very difficult. However, the approach has its merits and would benefit from further research. A method for verifying the liveness properties of the converter is presented which is used to construct a converter that possesses bounded queues and is deadlock and unspecified-reception-free.

A modification to Okumara's technique is proposed in [203] and an algorithm presented. Using CFSMs as a basis, the proposed algorithm first reduces the state transition graphs (STGs) for the two interworking protocols. Okamura's approach is then applied to generate the reduced CFSM model of the converter. It is the action of reducing the STGs that provides the scope to lower the computational complexity of Okumara's algorithm by one or two orders of magnitude without any detrimental effects on the resultant convertor. This pre-processing approach is capable of augmenting other methods, for example the interface equation work [218, 217] detailed

below; and suggests what could be a very promising line of research in carrying out this augmentation and examining the effect on computational complexity.

Another algorithmic approach is described in [180] which models interacting protocols as ordered mapping sets. The interesting aspect of this work is that it demonstrates that if two interworking protocols are safe, then the algorithm described in [180] guarantees that the converter will be safe and that a solution will be generated if it exists. The algorithm consists of six components and would appear automatable. However, while no formal analysis is available to prove otherwise, the algorithm would appear to be **NP** [94]. This area would benefit from further research, for example, work concerned with examining heuristics which can overcome the computational complexity problems.

More recently, Kakuda et al [117, 106, 118] have like Green and others [86, 126] attempted to synthesise protocol converter specifications automatically from the service specifications, but with the additional criteria of detecting unspecified errors due to collisions. The subsequent algorithm is based on the notions of prioritised primitives and projection. No indication is given as to the computational complexity of the algorithms, but it does appear to be a promising approach. The authors assert that their approach is more efficient than that of Saleh for instance [206, 205], which adopts a somewhat similar approach. Communication with Professor Kakuda indicates that research into the algorithm is continuing.

A succession of papers from Norris, Shields and Martin mark the progress of what is perhaps the most concerted effort to solve such equations in a general setting. Throughout their research, the base specification language is Milner's CCS [152]. While Norris formulated the problem [164], Shields provided a method of solution for the weakly determinate case by defining the important relations of I-completeness and O-completeness [217]. Shields later developed a completely general algorithm based

on this work [218, 219]. This was referred to as the discarding algorithm which is a 'brute-force' approach to the problem. In general, the algorithm produces maximal solutions, which can nevertheless be reduced using observational equivalence (essentially identifying τ-cycles and contracting them). Martin attempted to generalise the work of Shields in order deal with nondeterminism[140]. Like Shield's previous discarding, Martin's variant was still computationally intractable.

The approach adopted by Martin for the constructive algorithm was quite different [141, 138, 139, 142]. The work involved constructing an initial 'guess' and then adding (or removing) more structure as needed. While less general than the discarding algorithm, the advantage of this approach is that the solution is sometimes minimal. However, additional processing is needed and hence the problem of computational hardness is not dealt with. Indeed, to make the constructive algorithm completely general would require additional algorithms, in particular merge-split [144].

Parrow's approach [175], which is implemented within Concurrency Workbench, is based on an interactive, tableaux method using the bisimulation semantics of CCS. The $\mathcal{M}_q$ machine must be deterministic. From a practical viewpoint this restriction is a serious one, since in many, practical, concurrent systems some element of nondeterminism will be present. Parrow's work is certainly interesting from a theoretical perspective, and it appears that it was never intended to address industrial sized problems. The algorithm has been implemented on Concurrency Workbench [156] developed by the University of Edinburgh.

Calvert and Lam [33, 32], in what they call a 'top-down' approach, also define an interface equation. In terms of the general statement of the problem, it is close to the work of Norris et al, though the application domain is exclusively protocol conversion. More explicitly, they recognise the problem as being a quotient-type problem, but do not explore this characteristic to its full potential (which we do in this thesis).

Indeed, in order to synthesize the messages of the protocol converter a mapping is required from the given protocols which Calvert and Lam do not specify. Calvert and Lam also assert that the algorithmic solution to the general quotient problem is impossible and that the problem is equivalent to the halting problem. While a completely general solution is never likely to be polynomial, the author feels that this statement is too strong. This thesis will show that a large class of problems can be solved efficiently, and we identify in chapter 8 further work which could completely generalise the authors work. The work of Calvert and Lam is a creditable attempt and is closely related to the discarding algorithm, but like that algorithm, their approach is computationally hard and as a result must be seen as largely unsuccessful in that it does not progress Shields work, which was conceived in 1987.

The work of Rajagopal and Miller [187] is of interest in that it appears to be a hybrid of Lam's common images and Okumara's conversion seed. The resultant algorithm is conceptually similar to Wei *et al* [254, 253]. The crux of the technique, which uses CFSMs as the base language, is the formation of the Cartesian product of the input protocols which are then searched to produce finite-length traces that are then combined to form the converter. The converter needs to be formally verified and validated to check whether it is free from properties such as deadlock. Unfortunately, this work suffers from the fact that a specification of the message relationships between the two interacting protocols is required.

A synchronisation model approach is examined in [221, 40]. This formal model addresses two particular problems of protocol conversion: translation and synchronisation. The authors claim that the proposed method is low in space complexity and produces a converter development method which has a number of positive characteristics such as the ability to cope with concurrent processing, ease of debugging and efficient development. No real evidence is presented to back up this claim.

With such a wide range of techniques available, none of which are totally general, it would appear that the protocol engineers best hope is to develop a protocol conversion toolkit as described by Auerbach [6]. This issue is discussed again in chapter 7.

There are a number of important points that have emerged from research into protocol conversion:

- There are few tools which enable the software or hardware engineer to provide a specification of an interface given some input–output specifications. Those that do exist are inefficient (being **NP** or worse) and of limited applicability [33].

- There is little theory concerning the underlying mechanisms which are used for the automatic derivation of interface specifications. This could be a reason why the algorithms that have been developed are **NP**.

- There is an open question which concerns the utility of current communications specification languages with respect to protocol conversion. For example, we feel that SDL for instance, is too cumbersome a medium to be used for developing theories, techniques and tools which would enable the automatic generation of interface specifications. The issue is: can appropriate techniques be developed to include data in the interface specification at a sufficiently high-level of abstraction that the process of protocol derivation is not inefficient.

- There is a deep research question which is to do with the fact that we do not yet fully understand the inheritance properties of the interface specification, for example, can you assert that an interface specification does not contain undesirable properties such as deadlock when automatically generated from two specifications which are deadlock-free?

- Tools that have been developed are not based on practical notations such as LOTOS. Most tools that have been developed have been based on more abstract

notations such as CCS.

- Many of the tools that have been developed for protocol conversion have had as their aim the fully automatic derivation of interface specifications. As a consequence of this these tools have only really been found applicable to very small protocols of three orders of magnitude in size less than practical protocols. There is scope for examining the impact of heuristic, semi-automatic techniques which may scale up better.

- The issue of scalability is critical. Of the techniques that have been discussed, very few, if any, can claim to have been applied to industrial scale examples. Industrial-strength protocol conversion exercises which have been published (such as [64]) show no evidence of non-standard techniques being used.

3.9 Summary

This chapter has reviewed the use of formal methods in communications systems, not only those associated with current software engineering such as Z and CSP but also mathematical formalisms such as Petri nets which have a longer pedigree. One of the striking things which we think emerges from this review is that a large number of potential high-value uses for formal methods in protocol engineering.

What is also clear is that this area of application is bedevilled with the same problems that occur generally with formal methods: gaps in theory many of which have been outlined in this chapter and a comparative lack of concrete demonstrations that scaling-up to real-life problems is possible.

We regard the latter as the most serious. Few papers address the problems of applying formal notations to large-scale applications. It is not a subject totally ignored by researchers, for example, Lin and Liu [134] propose a three- fold strategy which

incorporates formal modelling, structural decomposition and functional abstraction and reachability analysis aimed at addressing the complexity issues encountered with industrial strength protocols. In particular, their method of structural decomposition and functional abstraction aids the process of breaking down a problem into manageable parts to which the reachability algorithm can then be applied without excessive computation problems. However, it is worth pointing out that as yet there are no reports of this strategy being employed on industrial strength protocols.

This lack of evidence of scalability bedevils the whole area, for example, many of the tools that have been developed for protocol conversion have had as their aim the fully automatic derivation of interface specifications. As a consequence of this, these tools have only really been found applicable to very small protocols of two to three orders of magnitude in size less than practical protocols. We are convinced that this is less of a problem than is perceived in the communications industry–certainly in the two areas detailed later in this section.

In the author's view the comparative lack of progress of formal methods in communications applications are those often cited for other areas: lack of trained staff, the problem of customer liaison and the high training costs.

There are a number of promising developments which affect formal methods which will increase their application in a number of areas, not just telecommunications applications.

First, there is the Integrated Methods approach being pioneered by various establishments. This approach combines standard structured methods such as SSADM and Yourdon Structured Development with formal specification languages such as VDM and Z [72, 91]. With this approach one uses formal languages only when absolutely necessary, otherwise the simpler structure methods are used.

This is eminently sensible: many communications systems which could benefit

from being developed formally contain large chunks of non-critical code which could be developed using less exact techniques.

Second, initiatives such as the British MoD's Defence Standard 00-55, which makes mandatory the use of formal methods for the critical components of a system may yet have an impact. Although intended for the defence industry, success is this area may provide incentives for industry in general.

Third, in the UK with mathematical methods of software development being on every computing curriculum, formal methods will become a more and more viable technology as graduates with the requisite skills trickle into industrial projects. La Trobe University has, for instance, developed an undergraduate protocol development training course based on formal methods [122]. It remains to be seen whether initiatives such as this will provide the necessary levers in terms of industrial applications.

Fourth, in the United States where university computing departments have traditionally had an arms length relationship to formal methods of development, a number of special issues of prestigious journals such as *IEEE Software*, *IEEE Transactions on Software Engineering* have devoted special issues to the subject.

However, the author feels that these positive aspects which have emerged over the last ten years are not enough to ensure the expansion in use of formal methods in the telecommunications industry. There is a clear need for some demonstration projects which are able to provide a high degree of assurance to the industrial communications community that real economic benefit accrues from formal methods. Of the areas surveyed in this chapter it is felt that this case can be made most easily in the areas of conformance testing and protocol conversion. Not only do we feel that this work contains scalable theories, but they do not require the use of the large numbers of trained personnel–something which has acted as a drag on the adoption of formal methods in large-scale software development.

Conformance testing within the OSI context using LOTOS is an encouraging area of work which is not to far away from being a practical proposition thanks to the work of Wezeman *et al* [246, 247]. Such testing is a massively expensive and repetitive task. The state of work in this area is such that we feel that an adequate theory has been produced and all that is necessary is for a case study which involves a real protocol.

The other area is protocol conversion. It is tempting to regard protocol conversion as a relatively unimportant area. However, computer communication systems permeate almost all corners of the globe. Yet, achieving useful communication between systems generally remains a non-trivial problem. Often this difficulty is due to the fact that the respective systems use different protocols, that is the syntax and semantics of the messages they send are generated by different sets of rules and procedures. When two systems are unable to communicate with each other a *protocol mismatch* is said to have occurred [86].

Currently developing hardware or software to mediate between two protocols is resource intensive task often carried out by electronic engineers using heuristic techniques. Typically, the specification of an interface between two real-life protocols can give rise to costs of the order of £250,000. British work, for example, on using formal notations such as CCS has suggested that this can be significantly reduced [140]. Other researchers, as has been discussed, have also attempted to derive automatic means of generating protocol converters. Typically however, these approaches are aimed at rather specific problem types and suffer from excessive computational complexity. The potential commercial gains and the nature of the technical problems suggested to the author that protocol conversion was a viable research topic. Further, the survey had suggested that of all the techniques developed, it was the work of Norris, Shields and Martin (the interface equation) which represented the best developed theory, and the one worthy of further investigation. As such, it formed the basis of

the research described in this thesis. The next chapter outlines this work.

Chapter 4

The Interface Equation

This chapter provides an overview of the interface equation and the underlying mechanisms associated with its solution. We initially adopt a wide perspective and examine a number of approaches which could be described as representing, although not explicitly, the general class of interface equation type theories. The focus however, is on the work of Norris, Shields and Martin, since it is the belief of the author that this represents the best developed theory for solving the equation to date. We review both the discarding and the constructive algorithms, and demonstrate that whilst the latter approaches a viable algorithm, the former has little practical significance. The chapter closes with a description of the authors research aims and objectives, which are based on the findings of this and previous chapters.

4.1 Introduction

We saw from chapter 3 that the use of formal methods in protocol engineering has been a rich area of study, particularly during the late nineteen eighties. It was also evident that protocol conversion is an ideal application for formal methods, which are

perceived to offer the best chance in terms of developing a completely general and concise *calculus of conversion.* While protocol conversion is generally treated as a distinct topic, it turns out that the problem of validation is similar in many ways to the conversion problem. This similarity of approach has been exploited by Chang and Liu in [40], where they transformed the conversion problem into a validation problem with relative ease (see chapter 3). In turn, verification techniques such as protocol projection have also been investigated. Hence, we find that the application of formal methods to protocol engineering is rich in both the respective domains of verification, validation and conversion, and the relationships between them.

The overriding problem it seems in terms of generating a grand unified theory, is the absence of a suitable wide-spectrum language. Even within the protocol conversion domain, no single formal notation has emerged as being sufficiently general for all problems, and as such, no single calculus of conversion theory has yet been established. Having said that, a number of theories have come close, most notably the work of Okumara, Liu, Lam, Green and Shields and Martin [168, 125, 218, 86, 33, 40, 221]. Of these, it is really the work of Norris, Shields and Martin which stands out as being the most complete theory of conversion synthesis [1]. This work also has detailed supporting literature and hence was the natural choice in terms of providing a basis for the research described in this thesis. This formal basis was important, since as we shall see later, and particularly in chapter 5, our method of solution is essentially topological in nature and requires a formal syntax and semantics. The work of Shields and Martin, which was initiated by Norris, is generally referred to as the *interface equation.*

As a result of the literature review the work described in this thesis adopted the protocol conversion problem as the key area for investigation. This was primarily to

[1] It should be noted however that the interface equation was not developed for protocol conversion — but for system construction.

satisfy the need to make the research practical. Secondly, it took the quotient nature of the problem, as in part identified by other researchers, as being central. The main emphasis of the work was to improve the process of protocol converter synthesis and to address the problem of scalability.

The next section investigates the discarding algorithm of Shields [217] and constructive algorithm of Martin [143]. A good understanding of these algorithms is an essential precursor to the research described in chapter 5.

4.2 Solving the interface equation

The term 'interface equation' was coined by Norris *et al* in 1985 [164]. These equations are also commonly referred to as 'context equations' [188]. The earliest reference which alludes to the practical application of these equations is given in [164], where they were seen to provide an ideal vehicle to address the problem of general system synthesis [13] and submodule construction. Indeed, the work emanated from the need to establish analytical techniques for a general theory of systems synthesis.

The equation encapsulates the general problem of concurrently operating systems which have to communicate via some interface. An interface equation is an equation of the form $(\mathcal{M}_p \mid \mathcal{M}_r) \setminus A \approx \mathcal{M}_q$, where $\mid$ is parallel composition, A the set of hidden actions (the events by which $\mathcal{M}_p$ and $\mathcal{M}_r$ communicate), and $\approx$ is some notion of equivalence. Machines [2] $\mathcal{M}_p$ and $\mathcal{M}_q$ are known, $\mathcal{M}_r$ is the interface machine to be found that satisfies the equation, if a solution exists.

The interface equation, in its various guises, can also be viewed in more general terms as an alternative representation of the *quotient machine* problem [3]. This notion of a quotient machine is intriguing since the anticipation is that the quotient

[2] We use the term machine in a general sense — it is synomonous with automata, process, system etc

machines structure and behaviour will be determined from the associated structure and behaviour of the two given machines, $\mathcal{M}_p$ and $\mathcal{M}_q$. The essence of the concept is that in the interface equation shown above, $\mathcal{M}_r$ can be expressed as the quotient of $\mathcal{M}_q$ over $\mathcal{M}_p$. In truth the situation is more complicated than this statement would seem to suggest. That the machines communicate with each other means that $\mathcal{M}_p$ cannot simply be 'factored out' from $\mathcal{M}_q$. The interface equation is formally defined as follows :-

4.2.1 Definition

An interface equation is an expression of the form

$$(\mathcal{M}_p|X)\backslash A \sim \mathcal{M}_q$$

where

1. $\Lambda(p) \cap \overline{\Lambda(q)} \subseteq \{\tau\}$ — the $\mathcal{M}_p$ and $\mathcal{M}_q$ machines cannot communicate.
2. $\Lambda(p) \cap \overline{A} = \emptyset$ — $\mathcal{M}_p$ has no actions in $\overline{A}$
3. $\Lambda(q) \cap (A \cup \overline{A}) = \emptyset$ — $\mathcal{M}_q$ has no actions in $A \cup \overline{A}$.

The resulting machine $\mathcal{M}_r$ must satisfy the following :-

4.2.2 Definition

$\mathcal{M}_r$ is a solution to the equation $(\mathcal{M}_p|X)\backslash A \sim \mathcal{M}_q \Leftrightarrow \mathcal{M}_r$ satisfies:

i) $(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \sim \mathcal{M}_q$

ii) $\Lambda(r) \cap \Lambda(p) \subseteq \{\tau\}$ — nontrivial actions of the $\mathcal{M}_p$ and $\mathcal{M}_r$ machines are distinct.

To see this assume that one has a solution to the interface equation $(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \approx \mathcal{M}_q$, let us denote its starting state by r. Then any sequence of transitions of the composite machine $(p \mid r)\backslash A$ to some arbitrary state $(p' \mid r')\backslash A$ must

be accompanied by a corresponding sequence of transitions from the q state to a state q' such that $(\mathcal{M}'_p \mid \mathcal{M}'_r)\backslash A \approx \mathcal{M}'_q$. More formally if $(p \mid r)\backslash A \Rightarrow^s (p' \mid r')\backslash A$, where s is some sequence of visible actions, then there must exist a state q' such that $q \Rightarrow^s q'$ and $(p' \mid r')\backslash A \approx q'$. Similarly, if $q \Rightarrow^s q'$ then there must exist states p' and r' such that $(p \mid r)\backslash A \Rightarrow^s (p' \mid r')\backslash A$ and $(p' \mid r')\backslash A \approx q'$. For a given state r' of the solution there may be several combinations of states of the known machines, p' and q', for which $(p' \mid r')\backslash A \approx q'$. If one collects together all such pairs then one can associate each state r' of the solution with a set of state pairs (p', q') such that $(p' \mid r')\backslash A \approx q'$. That is, for each state r' of the solution one can associate a set $K(r')$ given by $K(r') = \{(p', q') : (p' \mid r')\backslash A \approx q'\}$ and $(p \mid r)\backslash A \Rightarrow^s (p' \mid r')\backslash A$ and $q \Rightarrow^s q'$ and s is some sequence of observable actions. These $K(r)$-sets are important. They are discussed further later in this chapter, and will surface again in chapter 5.

Using the sets $K(r')$ for each state r' of the solution it is possible to define derivations between the sets in an analogous manner to derivations between the states themselves. These derivations and the notion of observational equivalence gives rise to the presence of certain conditions that these K-sets must satisfy individually and collectively in order that a solution to the interface equation exists – the so called I-completeness and O-completeness conditions (see Defs 4.3.2 and 4.3.3). If we denote $R(p)$ as the set of states reachable from p and $R(q)$, the set of states reachable from q, then the process of finding a solution can be expressed as the determination of a subset of the power set $P(R(p) \times R(q))$ which satisfy these I-completeness and O-completeness conditions [140].

4.3 The discarding and constructive algorithms

In this section we will examine the solution mechanism developed by Shields and Martin. There are a number of basic concepts which need to be covered before one

can attempt to understand the theory. The solution centres on the construction of special sets, the K-sets, which need to meet certain conditions (I-completeness and O-completeness) if a solution is to be found. K-sets are fundamental to the discarding and constructive algorithms. The internal structure of K-sets is intimately related to the structure of $\mathcal{M}_p$. In addition, each K-set is associated with one, and only one, state of $\mathcal{M}_r$. Each K-set consists of a set of triples (p, q, r). Since each K-set is associated with one and only one r, the r is usually dropped from the triple and attached to the K, so a distinct K-set is labeled $K(r)$. The K-sets partitions $R(p) \times R(q)$ ($R()$ is defined in 4.3.1) so all ordered pairs $(p, q) \in K(r)$ have the property that $(p|r)\backslash A \approx q$. K-sets are formally defined in definition 4.4.5.

4.3.1 Definition

Let $\mathcal{M}_p$ be a machine which can exist in the state p then $R(p)$ is defined to be the set of all states reachable from the state p. The set $\Lambda(p)$ is defined to be the set of all actions which are possible $\forall p' \in R(p)$. Let A be a set of actions then $R_A(p)$ is the set of all states reachable from p by sequences, s, of actions not contained in A; *i.e.* $s \in (\Lambda(p) - A)^*$.

$K(r)$-sets by themselves are not sufficient to generate a solution. Two new concepts, I-completeness and O-completeness, which were developed by Shields [217], provide the necessary and sufficient conditions to find a solution. This was an important step, which together with K-sets, provide the cornerstones for the theory. The definitions for I-completeness (Def 4.3.2) and O-completeness (Def 4.3.3) are given below.

4.3.2 Definition

K is said to be *I-complete* iff $\forall(p', q') \in K$, if $p' \rightarrow^{\mu} p''$ for some p'' where $\mu \notin A$ and $\mu \neq \tau$ then $q' \Rightarrow^{\mu} q''$ for some q''.

For O-completeness we need to consider transitions between K-sets (and hence between states in the $\mathcal{M}_r$ machine). There are two types since $\mathcal{M}_r$ will either do a transition which does not involve communication with $\mathcal{M}_p$ or does communicate (synchronise) with a transition in $\mathcal{M}_p$. We refer to these transitions as μ, O (where $\mu \in \Lambda(q) - \Lambda(p) - \{\tau\}$) and μ, C (where $\mu \in A$) respectively. The difference is that if we have a $K \rightarrow^{\mu,O} K'$ then every pair $(p, q) \in K$ will have a μ, O transition to a pair $(p', q') \in K'$. A μ, C transition, on the other hand, need only be between one element of K and one element of K' [3]. We can now define O-completeness.

4.3.3 Definition

K is said to be *O-complete w.r.t.* S, where S is a set of K-sets, iff $\forall(p', q') \in K$ if $\mu \in \Lambda(q)$ and $q' \rightarrow^{\mu} q''$ then $\exists m, n \geq 0$, $\mu_1 \ldots \mu_n$, $\mu'_1 \ldots \mu'_m \in A$, $p_0 \ldots p_n$, $p'_0 \ldots p'_m \in R(p)$ and $K_0 \ldots K_n$, $K'_0 \ldots K'_m \in S$ *s.t.*

i) $p' = p_0 \Rightarrow^{\mu_1} p_1 \Rightarrow^{\mu_2} \cdots \Rightarrow^{\mu_n} p_n$

ii) $K = K_0 \Rightarrow^{\overline{\mu}_1,C} K_1 \Rightarrow^{\overline{\mu}_2,C} \cdots \Rightarrow^{\overline{\mu}_n,C} K_n$

iii) Either

(a) $\mu \neq \tau$ and $p_n \Rightarrow^{\mu'} p_0$ and $K_n \Rightarrow^{\epsilon} K'_0$, or

(b) $\mu \neq \tau$ and $p_n \Rightarrow^{\epsilon} p'_0$ and $K_n \Rightarrow^{\mu',P} K'_0$, or

(c) $\mu = \tau$ and $p_n \Rightarrow^{\epsilon} p'_0$ and $K_n \Rightarrow^{\epsilon} K'_0$

[3] We will see in chapter 5, where we consider the topological properties of machines, that the nature of the μ, O and μ, C transitions is intimately linked to the structural symmetry of $(\mathcal{M}_p \mid \mathcal{M}_q)\backslash A$.

iv) $p'_0 \Rightarrow^{\overline{\mu}'_1} p'_1 \Rightarrow^{\overline{\mu}'_2} \cdots \Rightarrow^{\overline{\mu}'_m} p'_m$

v) $K_0 \Rightarrow^{\overline{\mu}',C} K_1 \Rightarrow^{\overline{\mu}'_2,C} K_2 \cdots \Rightarrow^{\overline{\mu}'_m,C} K'_m$

vi) $(p'_m, q'') \in K'_m$

The set S is referred to as a *system.* It is a set of K-sets which together form a potential solution. Informally, one can view S as a sort of cover over $R(p) \times R(q)$. The discarding algorithm starts with S as the set of all possible K-sets, some of which are discarded as the algorithm proceeds. The constructive algorithm adopts a more general definition of the K-sets and applies certain conditions which immediately rule out certain sets. In this case S has fewer, but larger K-sets. In general S will be to small and require additional points to be added — thus the term constructive [4]. In both cases however, the set S is gradually refined until the complete set satifies I-completeness and O-completeness ($\hat{I}$-completeness and $\hat{O}$-completeness for the constructive algorithm). These I-completeness and O-completeness relations are necessary and sufficient conditions for solution (if it exists). The essential point is that every state of the unknown machine can be expressed as a subset of the Cartesian product of the two known machines, subject to these I-completeness and O-completeness conditions. Intuitively, the I-completeness and O-completeness conditions amount to establishing a path of actions between any two ordered pairs in $R(p) \times R(q)$, which may be in different K-sets. A K-set is essentially I-complete if it is connected, i.e. there is a path connecting any two elements of K. But if the two pairs in question lie in different K-sets then we will need transitions from one K-set

[4] This highlights the distinction between the two approaches. The discarding algorithm starts with a very large S which gets smaller. The constructive algorithm starts with a small S which gets larger. Since the discarding algorithm starts from the 'top' and works 'down' it's solutions tend to be maximal. The constructive algorithm however starts from the 'bottom' and works 'up', so its solutions tend to be minimal.

to another to establish the path. Since each K-set corresponds to a unique state r of $\mathcal{M}_r$, then these transitions between the K-sets are in fact the transitions of $\mathcal{M}_r$. One can view I-completeness as being associated with the reachability (or connectedness) of the $\mathcal{M}_p$ machine, and O-completeness with the connectedness of the $\mathcal{M}_r$ machine. I-completeness says that a transition in $\mathcal{M}_q$ is equivalent to a sequence of transitions in $\mathcal{M}_p$ which lie entirely within one K-set. O-completeness says that we must add actions from $\mathcal{M}_r$ to find an equivalent sequence of transitions if $\mathcal{M}_p$ is unable to perform the appropriate action(s). If a set of K-sets have sufficient transitions so that any two pairs may be 'connected' then they are said to be I-complete and O-complete. Informally, what we have is a path searching algorithm which attempts to find a suitable path from one pair to another — if it gets to a point where it can go no further, it starts again and tries a different route. A K-set which 'blocks' any such path is said to compromise a potential solution 4.3.4 (we refer to a potential solution as a *system* and denote it by S). Thus a solution is said to have been found when we have a set of K-sets which is both I-complete and O-complete. Another way of looking at it is to say that a solution has been found when we have an uncompromised set of K-sets. This is summarised in figure 4.1.

4.3.4 Definition

Let S be a potential solution of the interface equation (a set of K-sets). We say that K *compromises* S iff K is either not I-complete or K is not O-complete *w.r.t.* S.

Let there be n_p states of $\mathcal{M}_p$ and n_q states of the $\mathcal{M}_q$ machine, then there will in general be $2^{n_p * n_q}$ different sets of such state pairs that one could create. Of these sets however only a few will correspond to states of the solution discussed above. The essential process of devising an algorithm to solve the interface equation involves finding those sets of the $2^{n_p * n_q}$ possible which can be identified with a solution. The

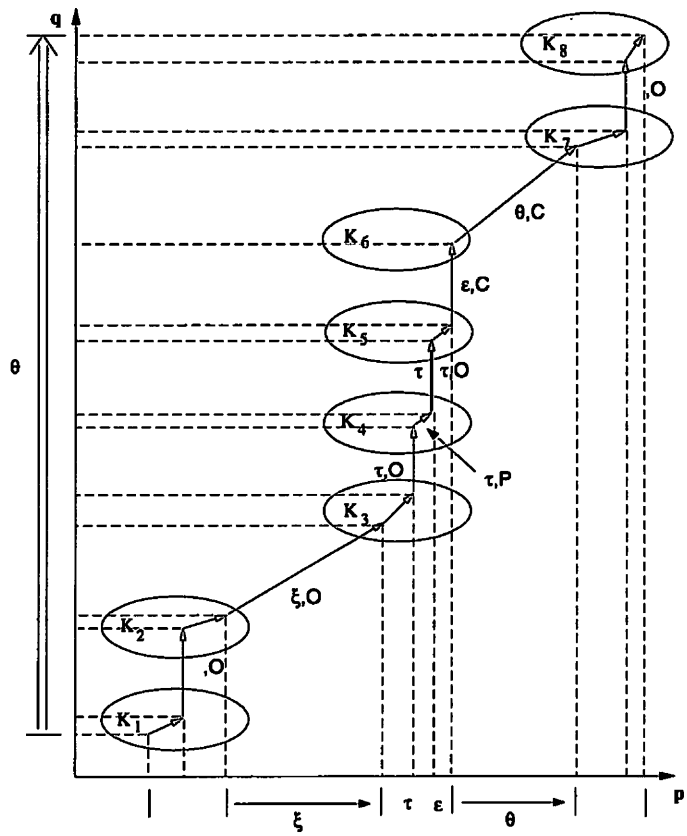


Figure 4.1: In this example $\mathcal{M}_q$ does an implicit θ transition from q to q'. $\mathcal{M}_p$ cannot carry out this transition, hence it must be done by $\mathcal{M}_r$. Here we see that a series of transitions, both within and between K-sets is needed to find an equivalent transition in $(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A$. In this way $(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \approx \mathcal{M}_q$ if this can be done for all transitions in $\mathcal{M}_q$. In other words, that the K-sets are both I-complete and O-complete.

process initially proposed by Shields [217], who developed the groundwork behind the approach for the weakly deterministic case, is to consider a system initially containing all such sets and then discarding them one by one if the required conditions — I-completeness and O-completeness — are not satisfied (for more detail on this algorithm refer to reference [217, 218, 219]). If during this process there remain no sets which contain the pair (p, q) (where p and q refer to the starting states of the known machines) then no solution is possible. Otherwise one will eventually obtain a system of K-sets all of which will satisfy the necessary conditions. It is then a simple step to assign each set a state name (with associated behaviour) which corresponds to a solution of the interface equation.

The main practical difficulty with the discarding algorithm is the huge number of sets that one needs to consider initially. Shields work was extended by determining the necessary and sufficient conditions for a solution in the general case (i.e. full non-determinism is accounted for) and devising an algorithm which operates in the 'opposite' way to the discarding algorithm. That is, a system is built of sets until we obtain a system which satisfies the required conditions. This is the constructive algorithm [143, 142]. By analysing the structure of the solution it is possible to rule out certain kinds of sets at a very early stage during the construction process which would have involved a large amount of effort in discarding algorithm.

The way the constructive algorithm works can be summarised as follows. Firstly the two machines are reduced to a minimal form where redundant states and actions, from the point of view of observational equivalence, are deleted. With the machines in such a form then any set corresponding to a solution state is such that for each state of $\mathcal{M}_p$, e.g. p', there is one and only one pair of the form (p', q') (where q' is some state of the reduced $\mathcal{M}_q$) in that set. Instead of generating all the sets of $P(R(p) \times R(q))$ two relations on the elements (p, q) of $R(p) \times R(q)$ are introduced

which must be satisfied by those elements to be part of a solution. These two relations are variants of the I-completeness and O-completeness conditions, and are referred to as $\hat{I}$-completeness and $\hat{O}$-completeness . These are formally defined as follows :-

4.3.5 Definition

Let $W \subseteq R(p) \times R(q)$ then W is said to be $\hat{\text{I}}$-*complete* iff $\forall (p', q') \in W$ and $\forall p''$ *s.t.* $p'' \in R_A(p')$ then $\exists (p'', q'') \in W$ *s.t.* $(p', q') \approx (p'', q'')$.

This set W is a sort of 'super K-set' (see below).

4.3.6 Definition

Given $S \subseteq P(R(p) \times R(q))$ then $(p', q') \in K' \in S$ (denoted by the triple (p', q', K')) is said to be $\hat{\text{O}}$-*complete* *in* S iff $\forall \mu \in q'$ and $q'' \in R(q)$ *s.t.* $q' \rightarrow^{\mu} q''$ $\exists n \geq 0$, $p_0, p_1 \ldots p_n, p_{n+1} \in R(p)$, $K_0, K_1 \ldots K_n, K_{n+1} \in S$ and $\mu_1 \ldots \mu_n \in \overline{A} \cup \{\tau\}$ *s.t.* $(p', q', K') \equiv (p_0, q', K_0) \rightarrow^{\mu_1} (p_1, q', K_1) \rightarrow^{\mu_2} (p_2, q', K_2) \cdots \rightarrow^{\mu_n} (p_n, q', K_n) \rightarrow^{\mu} (p_{n+1}, q'', K_{n+1})$

Notice that these are more general (weaker) than their respective counterparts. By applying these weaker conditions the resulting sets are in general quite large – we could call them 'super K-sets'. The elements that do not satisfy these conditions are discarded until a subset of $P(R(p) \times R(q))$ is found that satisfies the conditions, denote this by S'. S' is in fact a set of super K-sets. If S' does not contain an element $(p, q) \in K \in S'$ then no solution is possible. This step of the algorithm quickly deletes pairs (elements) which cannot form part of the solution. The next step involves using the equivalence class $\sim_{S'}$ (Def 4.3.7) which is defined over the $\hat{I}$-completeness condition to partition the set S'.

4.3.7 Definition

Let $(p', q'), (p'', q'') \in S \subseteq R(p) \times R(q)$ then define the relation $\approx_S$ by the following: $(p', q') \sim_S (p'', q'')$ iff

1. $p' \in R_A(p'')$ **or** $p'' \in R_A(p')$ **and**

2. - **if** $p' \in R_A(p'')$ **then** $L_{p'',p'} = \underline{L}_s((p'', q''), (p', q'))$, **if** $L_{p'',p'} > 1$ **then** $\forall \mu \in \Lambda(p) - A$ **if** $p'' \Rightarrow^{\mu} p'$ **then** $q'' \Rightarrow^{\mu} q'$, **and**
   - **if** $p'' \in R_A(p')$ **then** $L_{p',p''} = \underline{L}_s((p', q'), (p'', q''))$, **if** $L_{p',p''} > 1$ **then** $\forall \mu \in \Lambda(p) - A$ **if** $p' \Rightarrow^{\mu} p''$ **then** $q' \Rightarrow^{\mu} q''$ **and**
   - **if** $p' = p''$ **then** $q' = q''$ **and** $\forall \mu \in \Lambda(p) - A - \{\tau\}$ **if** $p' \Rightarrow^{\mu} p'$ **then** $q' \Rightarrow^{\mu} q'$.

where L is the length of a trace (of states) and $\underline{L}$ is the length of a (corresponding) sequence of actions.

4.3.8 Remark

The relation $\approx_S$ is symmetric, but not transitive or reflexive.

In order to generate an equivalence class, we need to find the transitive closure of $\approx_S$ (Def 4.3.9).

4.3.9 Definition

Define $S_{p,q}$ to be the largest subset of $R(p) \times R(q)$ which is $\hat{I}$-complete . Let $\sim_{S_{p,q}}$ be the transitive closure of $\approx_{S_{p,q}}$ (note that $\approx_{S_{p,q}}$ is a reflexive relation) then define $\phi_{\hat{I}}(p, q)$ to be the set of equivalence classes given by $\sim_{S_{p,q}}$ on $S_{p,q}$.

4.3.10 Remark

$\forall K \in \phi_{\hat{I}}(p, q)$ then K is $\hat{I}$-complete .

This partition gives the basic form of the sets which will form the solution, if it exists. $\hat{O}$-completeness is then applied over this partition. So the sequence is to first check for $\hat{I}$-completeness , generate $\sim_S$ and partition S to form K-sets, and then to check for $\hat{O}$-completeness .

As stated above, the K-sets generated by the discarding algorithm obey the I-completeness and O-completeness conditions which exist between the sets. In the constructive algorithm one uses a weaker version of these conditions so that sets (and hence the pairs that they contain) which clearly do not form part of the solution can be computed quickly and hence eliminated. This is an iterative process in that the deletion of a pair may affect the conditions of other pairs which would subsequently result in their deletion. The process is continued until all pairs satisfy the required conditions. Under certain circumstances a solution may be obtained already at this stage which involves polynomial time algorithms from graph theory. In general further processing is required in the form of an extraction process. This is achieved through a approach which is equivalent to a tree search with backtracking. Essentially, particular subsets are chosen in turn from the above partition, in general there will be a number of choices that can be made. After each subset is chosen some processing is required to ensure that the necessary conditions still hold. Eventually a situation is reached where no more subsets can be chosen. If the required conditions are not satisfied then the process is backtracked to choose another subset from the list of choices. The efficiency of this part of the algorithm is dependent on good heuristics to make the best choice. In the examples that have been considered it was found that fairly simple heuristics are sufficient to obtain a solution quite quickly. However, in the general case more sophisticated heuristic may be necessary (or the need for human intervention) in order to guide the system towards a potential solution. Notice that a fully extracted set of $\hat{I}$-complete and $\hat{O}$-complete K-sets is in fact I-complete and

O-complete, since I-completeness and O-completness are properties of the solution – not the method used to derive it.

If the algorithm reaches a point where no more sets can be chosen and all the conditions are satisfied then again in certain circumstances this situation results in a solution to the interface equation. This condition is : all states of $\mathcal{M}_p$ are reachable from any other state by sequences of actions not contained in A. For classes of interface equation that do not satisfy this condition then further processing may be required. A potential solution to this problem is given in [144, 143].

For certain classes of interface equation the constructive algorithm as currently implemented can obtain complete solutions far quicker than the discarding algorithm. Furthermore, the algorithm is more widely applicable than the method given in the concurrency workbench [175] in that nondeterminism in the machines can be taken into consideration. The current algorithm is particularly suited to solving interface equations which are such that each state of the p machine is reachable by actions not contained in A. In such cases the size of the sets obtained during the solution is well defined. In the more general case the algorithm requires a procedure for merging and splitting the sets during the construction procedure. This introduces further complexity into the solution procedure and would also greatly benefit from good heuristics [144].

This concludes the introductory material to the discarding and constructive algorithms. The interested reader is strongly recommended to study the references.

4.4 A small example

Having established an informal description of the interface equation, a small example can now be investigated. In this example we will use the discarding algorithm to generate a solution. This example is by necessity more rigorous than our previous

informal discussion. Consider two machines $\mathcal{M}_p$ and $\mathcal{M}_q$ given by the following agents.

$\mathcal{M}_p$ is given by:

$$\begin{aligned} p_1 &\Leftarrow \alpha p_1 + \alpha p_2 \\ p_2 &\Leftarrow \beta p_1 + \tau p_1 + \epsilon p_2 \end{aligned}$$

$\mathcal{M}_q$ is given by:

$$\begin{aligned} q_1 &\Leftarrow \alpha q_1 + \alpha q_2 + \mu q_3 \\ q_2 &\Leftarrow \beta q_1 + \tau q_1 + \mu q_4 + \tau q_4 \\ q_3 &\Leftarrow \alpha q_3 + \alpha q_4 + \nu q_1 \\ q_4 &\Leftarrow \beta q_3 + \tau q_3 + \nu q_2 + \tau q_2 \end{aligned}$$

Let A be given by $A = \{\epsilon\}$.

The problem is to find a machine M_r which has a state r_1 such that $(p_1 \mid r_1)\backslash A \approx q_1$. One first notes that $q_2 \approx q_4$ since there is a τ-cycle between these states. (A τ-cycle is a sequence of states $q'_1 \ldots q'_n : q'_1 \rightarrow^\tau q'_2 \rightarrow^\tau \cdots \rightarrow^\tau q'_n \rightarrow^\tau q'_1$.) Hence $\mathcal{M}_q$ can be reduced to a machine with observationally distinct states to obtain, after relabeling:

$$q'_1 \Leftarrow \alpha q'_1 + \alpha q'_2 + \mu q'_3 + \tau q'_3 \tag{4.1}$$

$$q'_2 \Leftarrow \beta q'_1 + \tau q'_1 + \mu q'_2 + \beta q'_3 + \tau q'_3 + \nu q'_2 \tag{4.2}$$

$$q'_3 \Leftarrow \alpha q'_3 + \alpha q'_2 + \nu q'_1 \tag{4.3}$$

Clearly $q_1 \approx q'_1$ and so we consider $\mathcal{M}_q$ to be given by 4.1–4.3. In the subsequent analysis the primes are dropped.

The solution, if it exists, will be an uncompromised system S of I-complete and O-complete K-sets.

The first step is to start constructing the K-sets, which we also require to be I-complete. Before doing this we need a few more definitions.

4.4.1 Definition

Let $(p',q'),(p'',q'') \in R(p) \times R(q)$, $\mu \in \Lambda(p) \cup \Lambda(q) - \{\tau\}$ and define

1. $(p',q') \rightarrow^{\mu,I} (p'',q'')$ iff $p' \rightarrow^{\mu} p''$ and $q' \Rightarrow^{\mu} q''$
2. $(p',q') \rightarrow^{\tau,P} (p'',q'')$ iff $p' \rightarrow^{\tau} p''$ and $q' \Rightarrow^{\epsilon} q''$ where ϵ is the null string
3. $(p',q') \rightarrow^{\tau,Q} (p'',q'')$ iff $q' \rightarrow^{\tau} q''$ with $q' \approx q''$ and $p' = p''$

4.4.2 Definition

Define

1. $I_{\mu,I}(p',q',p'') = \{(p'',q'')|(p',q') \rightarrow^{\mu,I} (p'',q'')\}$
2. $I_{\tau,P}(p',q',p'') = \{(p'',q'')|(p',q') \rightarrow^{\tau,P} (p'',q'')\}$

These sets define types of neighbourhood associated with each ordered pair.

4.4.3 Definition

Define $\Re^i(p',q')$ ($i \in \{1,2\ldots\}$) by the following:

1. $(p',q') \in \Re^i(p',q')$
2. if $(p'',q'') \in \Re^i(p',q')$ and $I_X(p'',q'',p''') \neq \emptyset$ (where $X \in \{(\mu,I),(\tau,P)\}$) then $I' \subset \Re^i(p',q')$ where I' is any non-empty subset of $I_X(p'',q'',p''')$ and $\forall(p''',q_i),(p''',q_j) \in I', i \neq j$ then $q_i \approx q_j$
3. if $(p'',q'') \in \Re^i(p',q')$ and $q''' \approx q''$ for some $q''' \in R(q)$ then $(p'',q''') \in \Re^i(p',q')$

In general for a given (p',q') there will be a number of $\Re^i(p',q')$ depending on which subsets are chosen in *ii*) of the above definition. The superscript i is used to distinguish between these subsets. When there is no danger of ambiguity, the superscript is dropped.

We give the set of all $\Re^i(p',q')$ a special name :-

4.4.4 Definition

Define $\psi(p,q) = \{\bigcup_{B \in X} \mid X \subseteq \{\Re^i(p',q') \mid (p',q') \in R(p) \times R(q)\}\}$

4.4.5 Definition

A K-set is an element of ψ.

4.4.6 Definition

Let $S \subseteq \psi$ be a system and $K \in S$ then $\Re(p,q) \subseteq K$. We define a *decomposition of* S *into* S' by the following conditions:

1. $\Re(p,q) \in S'$
2. if $K \rightarrow^{\mu,Y} K'$ exists for some K, $K' \in S$ and $L \subset K$ with $L \in S'$ *s.t.* $L \rightarrow^{\mu,Y} K'$ then the set given by the minimal union of $\Re$ from K' containing the image of the μ, Y derivation from L to K' is also contained in S'.
3. if $K \rightarrow^{\tau} K'$ exists for some K, $K' \in S$ and $L \subset K$ with $L \in S'$ then the set given by the minimal union of $\Re$ from K' containing the image of the τ derivation from L to K' is also contained in S'.

4.4.7 Definition

Let $(p',q') \in R(p) \times R(q)$. Define $(p',q') \rightarrow^{\mu,C} (p'',q'')$ iff $p' \rightarrow^{\overline{\mu}} p''$ and $q' \Rightarrow^{\epsilon} q''$ (where ϵ is the null string) with $\mu \in \overline{A}$. If $K, K' \in \psi(p,q)$ then define $K \rightarrow^{\mu,C} K'$ iff:

i) $\exists (p',q') \in K$ and $(p'',q'') \in K'$ *s.t.* $(p',q') \rightarrow^{\mu,C} (p'',q'')$

ii) $\forall (p',q') \in K$ if $p' \rightarrow^{\overline{\mu}} p''$ with $\overline{\mu} \in A$ then $\exists (p'',q'') \in K'$ *s.t.* $(p',q') \rightarrow^{\mu,C} (p'',q'')$.

We shall now obtain a system S which contains no compromising K-sets. By Definition 4.4.6 the system must contain a set $\Re(p_1, q_1)$ which we calculate first. By Definition 4.4.3 i) $(p_1, q_1) \in \Re(p_1, q_1)$ and by Definition 4.4.3 ii) we need to calculate all non-empty $I_X(p_1, q_1, p')$ where $X \in \{(\mu, I)(\tau, P)\}$, $p' \in R(p_1)$, $\mu \in \Lambda(p_1)$. These are $I_{\alpha,I}(p_1, q_1, p_1)$ and $I_{\alpha,I}(p_1, q_1, p_2)$ which are given, from Definition 4.4.7, by:

$$I_{\alpha,I}(p_1, q_1, p_1) = \{(p_1, q_1)(p_1, q_3)(p_1, q_2)\} \tag{4.4}$$

$$I_{\alpha,I}(p_1, q_1, p_2) = \{(p_2, q_1)(p_2, q_2)(p_2, q_3)\} \tag{4.5}$$

By way of an example we will look at the derivation of $I_{\alpha,I}(p_1, q_1, p_1)$ more closely. Referring to definitions 4.4.1 and 4.4.2 we see that we wish to find the set of pairs for which there is an α transition from (p_1, q_1) to (p_1, q'). Hence, we need to find the set of q-states reachable, via an α, from q_1. Referring to equations 4.1-4.3 it can be seen that $p_1 \rightarrow^{\alpha} p_2$ and that $q_1 \rightarrow^{\alpha} q_1$, $q_1 \rightarrow^{\alpha} q_2$, and $q_1 \rightarrow^{\tau} q_3 \rightarrow^{\alpha} q_3$ (or $q_1 \Rightarrow^{\alpha} q_3$). Thus we arrive at the set given in equation 4.4. The process is repeated for all I_X sets.

Now, for any $(p', q_1')(p', q_2') \in K \in S$ where S is an uncompromised system, then $q_1' \approx q_2'$. This follows since $(p'|r_s(K))\backslash A \approx q_1'$ and $(p'|r_s(K))\backslash A \approx q_2'$. Thus since $\Re(p_1, q_1)$ does not compromise the system and no pair of states given by 4.1–4.3 are observationally equivalent we can only have one element of 4.4 and 4.5 contained in $\Re(p_1, q_1)$. By Definition 4.4.3 $(p_1, q_1) \in \Re(p_1, q_1)$ and so we can only have $\Re(p_1, q_1) \cap I_{\alpha,I}(p_1, q_1, p_1) = \{(p_1, q_1)\}$. Considering Equation 4.5 we cannot have (p_2, q_1) nor (p_2, q_3) contained in $\Re(p_1, q_1)$ since by Definition 4.3.2 $\Re(p_1, q_1)$ would not then be I-complete. We thus have $\Re(p_1, q_1) \cap (I_{\alpha,I}(p_1, q_1, p_1) \cup I_{\alpha,I}(p_1, q_1, p_2)) = \{(p_1, q_1)(p_2, q_2)\}$. One must now consider the μ, I and τ, P derivations from (p_2, q_2), obtaining:

$$I_{\beta,I}(p_2, q_2, p_1) = \{(p_1, q_1)(p_1, q_3)\} \tag{4.6}$$

$$I_{\tau,P}(p_2, q_2, p_1) = \{(p_1, q_1)(p_1, q_2)(p_1, q_3)\} \tag{4.7}$$

Using the same arguments as for Equation 4.4 one can only have $\Re(p_1, q_1) \cap I_{\beta,I}(p_2, q_2, p_1) = \Re(p_1, q_1) \cap I_{\tau,P}(p_2, q_2, p_1) = \{(p_1, q_1)\}$. Thus for $\Re(p_1, q_1)$ we have the only choice:

$$\Re^{(1)}(p_1, q_1) = \{(p_1, q_1)(p_2, q_2)\}$$

This is certainly I-complete and so we now need to find sets to add to our system to make it O-complete.

First consider $(p_1, q_1) \in \Re^{(1)}(p_1, q_1)$. Clearly $\Re^{(1)}(p_1, q_1)$ is O-complete with regards the $q_1 \rightarrow^{\alpha} q_1$ and the $q_1 \rightarrow^{\alpha} q_2$ derivations. The $q_1 \rightarrow^{\mu} q_3$ derivation indicates that a μ, O derivation is required since $\mu \notin \Lambda(p)$. Conditions 1,2,4 and 5 in Definition 4.3.3 obviously do not apply since $p_1 \not\Rightarrow^{\phi}$ where $\phi \in A$. Also since $\mu \notin \Lambda(p)$ then we must have condition $3b$). We thus consider possible τ and μ, O derivations from $\Re^{(1)}(p_1, q_1)$. The images possible of a μ, O derivation from $\Re(p_1, q_1)$ are $\{(p_1, q_3)[(p_2, q_2)(p_2, q_3)]\}$ where we use the notation [ ] to denote any non-empty subset of the elements contained within [ ]. Now (p_2, q_3) cannot be contained within this set because it would not then be I-complete (since $p_2 \rightarrow^{\beta}$ but $q_3 \not\Rightarrow^{\beta}$) and so the only image possible is $\{(p_1, q_3)(p_2, q_2)\}$. Next we calculate $\Re(p_1, q_3)$ and $\Re(p_2, q_2)$. One obtains:

$$I_{\alpha,I}(p_1, q_3, p_1) = \{(p_1, q_3)(p_1, q_2)(p_1, q_1)\} \tag{4.8}$$

$$I_{\alpha,I}(p_1, q_3, p_2) = \{(p_2, q_3)(p_2, q_2)(p_2, q_1)\} \tag{4.9}$$

$$I_{\beta,I}(p_2, q_2, p_1) = \{(p_1, q_1)(p_1, q_3)\} \tag{4.10}$$

$$I_{\tau,P}(p_2, q_2, p_1) = \{(p_1, q_2)(p_1, q_1)(p_1, q_3)\} \tag{4.11}$$

Since we require $\Re(p_1, q_3)$ to be I-complete then $\Re(p_1, q_3) \cap I_{\alpha,I}(p_1, q_3, p_2) = \{(p_2, q_2)\}$. Also since our required set, which is of the form $\Re(p_1, q_3) \cup \Re(p_2, q_2)$, contains (p_1, q_3) and no pairs of q_1, q_2, and q_3 are observationally equivalent to each other then the only elements that can be considered for the set from Equations 4.8–4.11 are

$\{(p_1, q_3), (p_2, q_2)\}$. If we identify this set with r_2 and $\Re^{(1)}(p_1, q_1)$ with r_1, *i.e.*:

$$r_1 = r_s(K_1) \quad \text{where} \quad K_1 = \{(p_1, q_1)(p_2, q_2)\} \tag{4.12}$$

$$r_2 = r_s(K_2) \quad \text{where} \quad K_2 = \{(p_1, q_3)(p_2, q_2)\} \tag{4.13}$$

we have $K_1 \rightarrow^{\mu} K_2$ ($r_1 \rightarrow^{\mu} r_2$). K_2 satisfies the O-completeness of K_1 for the derivation $q_1 \rightarrow^{\mu} q_3$. The final derivation from q_1 that we need to consider is $q_1 \rightarrow^{\tau} q_3$. To satisfy O-completeness we require (note p_1 has no τ actions or actions contained in A) $K_1 \Rightarrow^{\epsilon} K_3$ no τ actions or actions contained in A) $K_1 \Rightarrow^{\epsilon} K_3$ with $(p_1, q_3) \in K_3$. The images possible of a τ derivation from $\Re(p_1, q_1)$ are given by $\{[(p_1, q_1)(p_1, q_3)][(p_2, q_2), (p_2, q_3)(p_2, q_1)]\}$. Again I-completeness allows only (p_2, q_2) in the second square brackets. Thus we can satisfy O-completeness for the $q_1 \rightarrow^{\tau} q_3$ derivation ($K_1 \rightarrow^{\tau} K_1$ gives no new information) only if we have $K_3 = K_2$ and so we have $r_1 \rightarrow^{\tau} r_2$.

We now need to look at $(p_2, q_2) \in K_1$ and consider derivations from q_2. They are: 1) $q_2 \rightarrow^{\beta} q_1$ 2) $q_2 \rightarrow^{\tau} q_1$ 3) $q_2 \rightarrow^{\mu} q_2$ 4) $q_2 \rightarrow^{\beta} q_3$ 5) $q_2 \rightarrow^{\tau} q_3$ 6) $q_2 \rightarrow^{\nu} q_2$. Taking cases 1)–5) in turn we show that O-completeness is already satisfied.

1. $p_2 \rightarrow^{\beta} p_1$ and $(p_1, q_1) \in K_1$

2. $p_2 \rightarrow^{\tau} p_1$ and $(p_1, q_1) \in K_1$

3. In this case we have $K_1 \Rightarrow^{\mu, O} K_2$ and $(p_2, q_2) \in K_2$

4. Here we only have $p_2 \Rightarrow^{\beta} p_1$ and $(p_1, q_3) \notin K_1$. However, O-completeness is satisfied because we have $p_2 \rightarrow^{\tau} p_1$, $K_1 \rightarrow^{\tau} K_2$ and $(p_1, q_3) \in K_2$.

5. Here O-completeness is satisfied because $p_2 \rightarrow^{\tau} p_1$, $K_1 \rightarrow^{\tau} K_2$ and $(p_1, q_3) \in r_2$

6. Here we clearly need a ν, O derivation at some stage. It is not possible to have $K_1 \rightarrow^{\nu, O}$ because $q_1 \not\Rightarrow^{\nu}$ but K_2 does have a ν, O derivation, the images of which

are $\{[(p_1,q_1)(p_1,q_3)],[(p_2,q_2)(p_2,q_3)(p_2,q_1)]\}$. For reasons stated previously due to I-completeness only the (p_2,q_2) term in the second set of square brackets leads to an uncompromised system. Considering the options in the first set of square brackets one has the choice $K_2 \rightarrow^{\nu,O} K_2$ or $K_2 \rightarrow^{\nu,O} K_1$. Both these choices satisfy the O-completeness condition for case 6). That is, if $K_2 \rightarrow^{\nu,O} K_2$ then we have $K_1 \rightarrow^{\tau} K_2 \rightarrow^{\nu} K_2$ and $(p_2,q_2) \in K_2$ or if $K_2 \rightarrow^{\nu,O} K_1$ then we have $K_1 \rightarrow^{\tau} K_2 \rightarrow^{\nu} K_1$ and $(p_2,q_2) \in K_1$.

Having satisfied the O-completeness of K_1 we now need to look at K_2. Clearly, K_2 is I-complete. The element (p_1,q_3) satisfies O-completeness using the sets and derivations already generated provided we have $K_2 \rightarrow^{\nu} K_1$. The agent q_2 in the second element has the derivations listed 1–6 above and the first of these does not satisfy the O-completeness condition with the results obtained so far. We therefore require further derivations from K_2. The only possible τ derivations from K_2 is $K_2 \rightarrow^{\tau} K_2$ which doesn't help and so an $\bar{\epsilon},C$ derivation is required. The images of an $\bar{\epsilon},C$ derivation from K_2 are given by $\{[(p_2,q_1)(p_2,q_2)(p_2,q_3)]\}$. We cannot have a set containing (p_2,q_1) because it would not then be I-complete and so we are left with two possible choices: $K_2 \rightarrow^{\bar{\epsilon},C} K_1$ and $K_2 \rightarrow^{\bar{\epsilon},C} K_2$. The $K_2 \rightarrow^{\bar{\epsilon},C} K_2$ derivation does not help since it leaves us in the same position. However, the $K_2 \rightarrow^{\bar{\epsilon},C} K_1$ derivation allows us to satisfy O-completeness for the $q_2 \rightarrow^{\beta} q_1$ derivation since $p_2 \rightarrow^{\epsilon} p_2 \rightarrow^{\beta} p_1$ and $(p_1,q_1) \in K_1$. We thus require $r_2 \rightarrow^{\bar{\epsilon}} r_1$. In fact, since $p_2 \rightarrow^{\epsilon} p_2$, $K_2 \rightarrow^{\bar{\epsilon},C} K_1$, $(p_2,q_2) \in K_1$ and (p_2,q_2) satisfy the O-completeness conditions in K_1 then (p_2,q_2) satisfies the O-completeness conditions in r_2. We therefore have that the system consisting of the sets K_1 and K_2 contains sets which are both I- and O-complete and since $(p_1,q_1) \in K_1$, then it constitutes an uncompromised system.

The transitions between the K-sets are thus :-

$$K_1 \rightarrow^{\mu} K_2$$

$$
\begin{aligned}
K_1 &\rightarrow^{\tau} K_2 \\
K_2 &\rightarrow^{\nu} K_1 \\
K_2 &\rightarrow^{\overline{\epsilon}} K_1 \text{ or} \\
K_2 &\rightarrow^{\nu} K_2 \\
K_2 &\rightarrow^{\nu} K_1 \\
K_2 &\rightarrow^{\overline{\epsilon}} K_1 \text{ or} \\
K_2 &\rightarrow^{\nu} K_1 \\
K_2 &\rightarrow^{\nu} K_2 \\
K_2 &\rightarrow^{\overline{\epsilon}} K_1 \\
K_2 &\rightarrow^{\overline{\epsilon}} K_2
\end{aligned}
$$

These translate immediately to the following $\mathcal{M}_r$ machines.

$$r_1 = \mu r_2 + \tau r_2 \tag{4.14}$$

$$r_2 = \nu r_1 + \overline{\epsilon} r_1 \tag{4.15}$$

$$r_1 = \mu r_2 + \tau r_2 \tag{4.16}$$

$$r_2 = \nu r_2 + \overline{\epsilon} r_1 + \nu r_1 \tag{4.17}$$

$$r_1 = \mu r_2 + \tau r_2 \tag{4.18}$$

$$r_2 = \nu r_1 + \overline{\epsilon} r_1 + \overline{\epsilon} r_2 + \nu r_2 \tag{4.19}$$

The associated graphs of these machines can be found in figure 4.2. It so happens that the r_1 in each of these generated solutions are observationally equivalent although in general this may not necessarily be so for multiple solutions.

Even with this small example, the details associated with the underlying theory are quite complex. It is perhaps not surprising that the algorithmic complexity of this approach is certainly **NP** and possibly **EXP**.

A similar example for the constructive algorithm can be found in [143].

4.5 Conclusion

This chapter has presented an overview of the interface equation. By presenting the basic theory we have laid the foundations for the work that follows — all the basic tools required have been covered. The simple example presented in this chapter demonstrates the principles involved. As can be seen the underlying approach is one of set generation and path searching. As such, it is perhaps not surprising that the complexity of the problem increases exponentially with the number of states of the respective machines. The discarding algorithm, being very much concerned with the behavioural (dynamic) properties of the machines, ignores many of the structural (static) clues inherent within the problem. It is a 'brute-force' approach which considers all possibilities, discarding those combinations which do not lead to a solution [5]. Hence, whilst it is of theoretical value, from a practical stand point it is not a viable option. To demonstrate this, a C program implementation of the algorithm took 16hrs of VAX processor time to solve a problem which contained a 4-state $\mathcal{M}_p$ and 20-state $\mathcal{M}_q$. Even accounting for improvements in processor capability (the example was done in 1987), it would take the discarding algorithm $\sim 2^{24220}$ days to solve the example given in appendix C. While Martin was able to address the limitation of weak determinacy and hence generalise the discarding algorithm, the problems of computational complexity, reachability and scalability remained.

[5] Note that the machine resulting from the Cartesian product of all states (along with transitions) will, if a solution exists, itself be a solution.

The constructive algorithm goes some way towards addressing the weaknesses of the discarding algorithm by including some aspects of structure, primarily in the derivation of the K-sets. If $\mathcal{M}_p$ is reachable by actions not in A, then the algorithm can have a considerable impact on the processing times, perhaps halving the time required by the discarding algorithm. However, if the reachability condition is not met then the algorithm could actually be worse than the discarding approach. This weakness could be addressed via a suitable merge-splitting algorithm, but this itself is likely to be computationally hard.

The discarding algorithm then presents an essentially behavioural approach to the problem of solving the interface equation. By considering some of the structural properties within the protocols, the constructive algorithm is, for a specific class of problems, a significant improvement. The research presented in this thesis has shown that adopting a largely structural viewpoint, much more so than the constructive algorithm, offers many advantages.

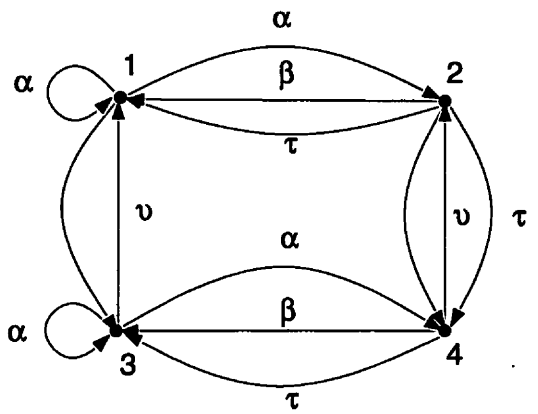
Referring back to the example, graphically speaking the structures of $\mathcal{M}_r$, $\mathcal{M}_p$ and $\mathcal{M}_q$ can be seen to be related in that they are 'almost' quotient graphs as indicated in figure 4.2.

This approach is exploited extensively in the next chapter, where we develop a theory based on the quotient graph concept. This is seen as a natural progression from the work of Shields and Martin, which although cumbersome, still represents the state of the art in the solution of the interface equation.

In summary then, the author's research set out to address the following weaknesses in the discarding and constructive algorithms and other approaches :-

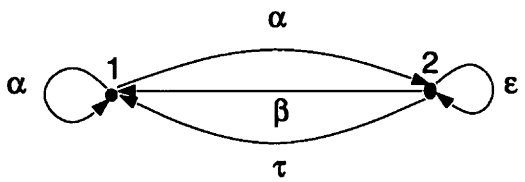
1. To address the problem of computational complexity. While the discarding algorithm is theoretically completely general, from a practical point of view the algorithm is unusable.

Mq



Divided by

Mp



is similar to

Mr

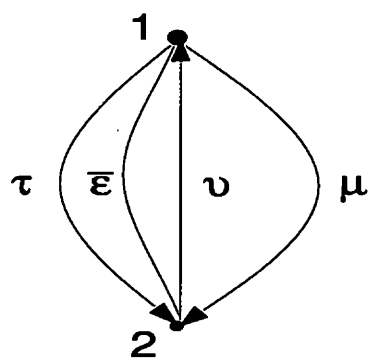


Figure 4.2: The graphs of $\mathcal{M}_r$, $\mathcal{M}_p$ and $\mathcal{M}_q$

2. To overcome the problems imposed by behavioural anomalies in the respective machines such as deadlock, livelock etc.

3. To overcome the limitations imposed by p-reachability, which amounts to solving problems with disconnected machine components.

4. To develop a technique which is able to identify quickly whether a solution is possible for the majority of cases.

5. Address the issue of scalability.

6. To broaden the class of solvable problems.

The theory developed as a consequence of these objectives is described in detail in the next chapter.

Chapter 5

A Graph Theoretic Approach

The application of quotient graph theory to the solution of the interface equation is presented. Starting with the establishment of the necessary links between CCS and graph theory, the chapter goes on to introduce and describe the concepts of τ-splitting, symmetric closure and quotient extraction. A number of theorems, corollaries, and lemmas are stated and proved. This is followed by the presentation of a selection of validation experiments, one of which is a large scale example. This validation process demonstrates that the algorithm can not only significantly improve processing times, but also overcomes a number of other weaknesses of previous approaches such as p-reachability. The merging of automata (via CCS) and graph theory also provides the catalyst for some interesting extensions to CCS. The concept of machine symmetry permeates throughout this process. Indeed, the notion of symmetry turns out to be a powerful concept, not only in terms of the algorithm itself, but also as a validation tool. We also identify areas where further research is required. The chapter closes with a brief discussion of the work.

5.1 Introduction

In this chapter we describe a procedure to solve CCS interface equations using graph theory. The approach is to represent generalised machines in the language of graphs, and to perform operations and transformations on these graphs. The decision to approach the problem in this way stemmed from the observation that we are essentially, as indicated in chapters 3 and 4, dealing with a quotient machine problem. The inference being that the $\mathcal{M}_r$ machine is somehow related to $\mathcal{M}_q$ modulo $\mathcal{M}_p$. Chapter 4 examined the development of the interface equation and the methods to solve it. It was asserted that each method, whilst not being explicitly stated by the respective researcher, involved a notion of machine quotient. Shields for instance recognised the problem to be akin to quotient-like construction [217]. The research described here has taken this concept to its next logical step by transferring the problem to a domain which has a rich body of theory concerning quotient structures — namely graph theory itself. In particular, a number of algorithms exist which can solve quotient graph problems in polynomial time (see section 5.2.1). Given that all previous methods of solving the interface equation are at best **NP**-complete, there was clearly an opportunity to reduce, perhaps radically, the computational complexity of these algorithms. While computational complexity was not the only concern — there were a number of other pressing problems (summary chapter 4) which the resultant theory would have to address — it did provide the initial focus for the research.

The research described here has demonstrated that substantial improvements can indeed be gained for a large class of interface equation problems. For the large case study in appendix C it is estimated that the graph theoretic approach reduced the solution time by approximately 2.5 orders of magnitude (chapter 7).

The chapter opens by establishing a firm link between the machine domain (as represented by CCS) and graph theory. Tau splitting is then discussed in detail. This

is an important part of the theory which enables the quotient algorithms to be applied. We then discuss symmetric closure, a technique which takes strongly equivalent interface equations and converts them to their structurally equivalent counterpart. A complete procedure for solving the interface equation is given in section 5.5, along with an example to demonstrate application. The chapter closes with a summary and discussion.

Note — As a note to the reader, we must stress that our use of CCS is at times non-standard. The reasons for this is that the graph approach requires certain new concepts in CCS which are defined in chapter 6. The reader may wish to read through that chapter before proceeding here.

5.2 Quotients in computer science

This section looks at the notion of machine quotient in more detail, and attempts to place the concept, at least in an abstract sense, firmly within the domain of computer science. The general notion of quotient exists in many forms within the computer science and mathematics disciplines. We show how other definitions within computer science turn out to be closely related to the notion of quotient machine.

The *idea* of a quotient machine has appeared on numerous occasions within this thesis, but without any formal definition being given. In this section a formal definition is given. But before doing so, let us examine at an abstract level what we mean by quotient machine. As with the preceding chapters of this thesis, we will use protocols and graphs as a specific examples.

We are given two protocols P_1 and P_2 which in terms of service provision ($S(P_1)$ and $S(P_2)$) have some commonality — that is the protocol projections [126] map on to a non-empty subset of the service domain ($S(P_1) \cup S(P_2)$). Our task is to derive $\frac{P_1}{P_2}$ or $\frac{P_2}{P_1}$. To make the discussion a little easier, we will now move to the graph domain

and examine the state transition graphs of these protocols — call them $\mathcal{G}_{P_1}$ and $\mathcal{G}_{P_2}$. We know from standard graph theory that the quotient will either exist or it will not, depending on whether the two graphs are coprime or not. As a analogy (the connection is in fact quite deep), a number m is a factor of a number n if it is in some sense contained within n. The same is true of graphs, and it is here that the service projection idea discussed in chapter 3 is useful. For the quotient protocols to exist (and this is a necessary but not sufficient condition) either $P_1 \subseteq P_2$ or $P_2 \subseteq P_1$. Let us assume that $P_1 \subseteq P_2$ and that we wish to find $\frac{P_2}{P_1}$. Then the quotient says 'take out of P_2 everything that P_1 does' the remaining structure being the quotient protocol. It is a simple step to generalise this to the more abstract notion of a machine, but these machines will need to exhibit similar (structural or behavioural) properties for the quotient to exist. Before giving a formal definition, let us look at a few examples of other quotient structures within computer science.

5.2.1 Quotient languages

Formal language theory is a deep and rich field. The aspect discussed here is but one example of a quotient. For more details consult [104].

5.2.2 Definition

Given two languages $\mathcal{L}_1$ and $\mathcal{L}_2$, the quotient language $\mathcal{L}^\circ \cong \frac{\mathcal{L}_1}{\mathcal{L}_2} = \{\zeta : \exists \eta \in \mathcal{L}_2 \text{ and } \zeta\eta \in \mathcal{L}_1\}$.

For example, let $\mathcal{L}_1 = \alpha^*\beta\alpha^*$ and $\mathcal{L}_2 = \alpha^*\beta$. For any string $\zeta \in \alpha^*$ we may choose $\eta = \beta$. Clearly $\zeta\eta \in \mathcal{L}_1$, hence the quotient $\mathcal{L}^\circ = \alpha^*$. We can see immediately the same qualitative mechanism as discussed with the protocols — indeed as protocols are languages themselves we might expect to see a close relationship between quotient languages and quotient machines (protocols).

5.2.3 Quotient FSMs

Finite state machine quotients depend on partitioning the respective machines in a certain way. Again, this is similar in many ways to partitioning a protocol with respect to its service specification [168].

5.2.4 Definition

Let M= (X, I, O, δ, β) be a finite state machine where X is a finite set of states, I a finite set of inputs, O a finite set of outputs, $\delta : X \times I \longrightarrow X$ (transition function), and $\beta : X \times I \longrightarrow O$ (output function). Let π be a partition of the state space S which has the substitution property SP and is output consistent [216], then the quotient finite state machine of M by π, written $\frac{\mathbf{M}}{\pi}$ is the machine $(\pi, I, O, \delta_\pi, \beta_\pi)$, where :-

$$\delta_\pi(X, i) = Y \quad \text{if and only if} \quad \delta(X, i) \subseteq Y$$
$$\beta_\pi(X, i) = o \quad \text{if and only if} \quad \beta(X, i) = o$$

5.2.5 Quotient machines

We now turn our attention to the generalised machines which will be the subject of this thesis. The following definition is new. The author has been unable to locate any definition of machine quotients. As can be seen, it is similar in many respects to the previous two quotient definitions, particularly the language quotient definition. But as the two previous quotient structures demonstrate (via the protocol link), we expect the various definitions to be identical in terms of semantics, if not in syntax.

5.2.6 Definition

Given two machines $\mathcal{M}_p$ and $\mathcal{M}_q$, then the quotient machine $\mathcal{M}_r = \frac{\mathcal{M}_q}{\mathcal{M}_p}$ is defined as the set of states and transitions $\{r \rightarrow^\alpha r' : \alpha \in \Lambda(q) - \Lambda(p) \wedge \exists p \rightarrow^\beta p' \in \mathcal{M}_p \bullet \beta \in \Lambda(p) \wedge (r \mid p), (r' \mid p') \in \mathcal{M}_q\}$

While expressed in the language of CCS, this is in fact a general definition. The problem then, is to map this notion of a quotient machine to the domain of quotient graphs. Before embarking on this, a survey of current graph quotient algorithms is given.

5.2.7 Quotient algorithms

In this section we very briefly survey the various approaches to deriving quotient graphs, and the mechanics associated with each algorithm. We also suggest a preferred approach with respect to solving the interface equation using CCS. This last point is important since converting the CCS specifications into an appropriate form is clearly a key step. Hence, the more natural the underlying constructs, be they incidence matrices, adjacency matrices or some other form, the more efficient the overall process is likely to be. A quick glance at a typical CCS, prefix notation, specification will show that there is a natural representation in matrix form.

Before doing so, a formal definition of quotient graph is given.

5.2.8 Definition

Let Γ be a group which operates on a set S. Then the *orbit* $\langle s \rangle$ of $s \in S$ under Γ is a subset of S consisting of all elements γs where $\gamma \in \Gamma$.

5.2.9 Definition

A *quotient space* is the topological space X which is the set of equivalence classes relative to some given equivalence relation R on a given topological space Y and is denoted Y/R. The topology of X is canonically constructed from that of Y.

We now combine the concepts of orbit and quotient space and define the quotient graph.

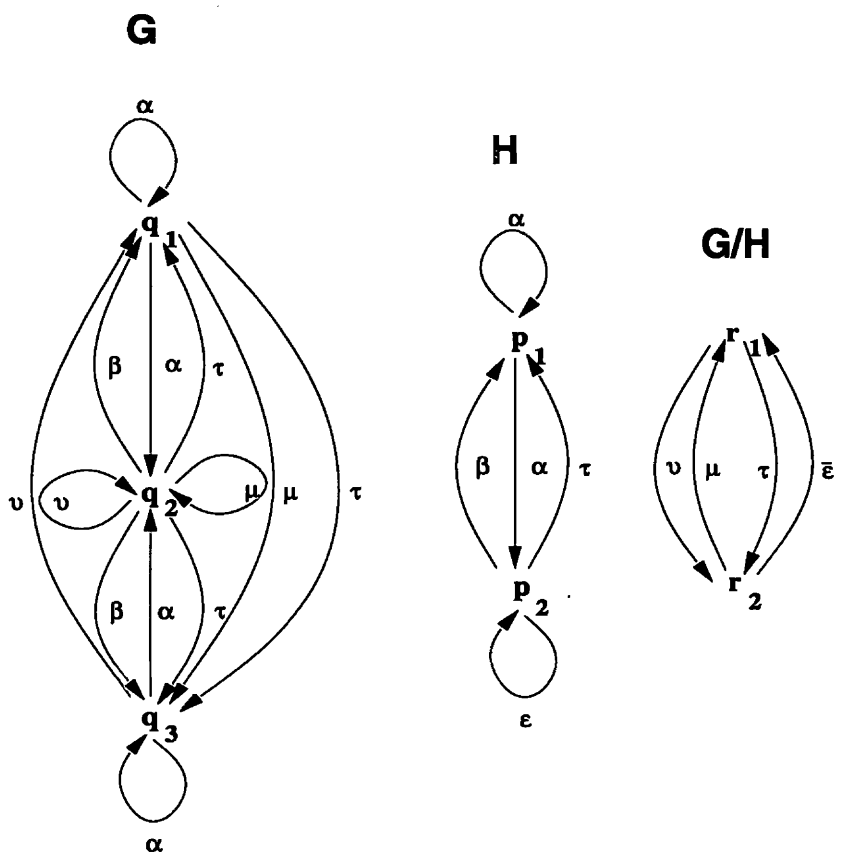


Figure 5.1: Illustration of the quotient graph mechanism.

5.2.10 Definition

Let G and H be labeled digraphs. Then the *regular quotient* G/H is the graph with vertex set $\mathcal{V} = \{G(V)/H(V)\}$ and edge set $\mathcal{E} = \{G(E)/H(E)\}$: the vertex orbit $\langle v \rangle$ is an end-point of the edge orbit $\langle e \rangle$ if any $v_i \in \langle v \rangle$ is an end-point of $e_j \in \langle e \rangle$.

It is easy to verify that each edge orbit has one or two vertex orbits as end-points [9]. Figure 5.1 illustrates how this definition works in practice. This definition of a quotient graph is the inverse of the Graph Cartesian Product (GCP) (see definition 5.3.1), i.e. $G \cong (G/H) \times H$.

The (polynomial) prime decomposition of a graph has been addressed by numerous

researchers [70, 69, 7, 107] etc. Winkler [250] for example, presents a method for deciding whether a given graph is a non-trivial Cartesian product. This is useful, if $\mathcal{M}_q$, once it had been processed via τ-splitting and unfolding, is prime, and $\mathcal{M}_p$ itself were non-trivial, then we know that no $\mathcal{M}_r$ exists — a useful check before applying the quotient algorithm itself. Indeed, one can envisage any tool implementation of the quotient approach using Winkler's approach as an initial test. The input form is the adjacency matrix — which is easily derivable from the CCS equations.

There are at least two of the polynomial-time algorithms for the undirected case which can be extended to the directed case. One of the first two polynomial-time algorithms for undirected factoring was given by Feigenbaum, Hershberger, and Schaffer [70]. How to extend the this algorithm to the directed case was shown in a paper by Feigenbaum [69]. Feigenbaum presents a polynomial algorithm for factoring weakly connected Cartesian product digraphs. An algorithm is presented which is readily transformed into computer code. On the face of it, the factorisation problem is not the one we want to solve. However, in the absence of anything else, these algorithms are readily usable in the quotient graph context. The process would be to first factorise $\mathcal{M}_p$ into its prime factors, and then $\mathcal{M}_q$. It is then a simple process to divide $\mathcal{M}_q$ by $\mathcal{M}_p$, with the remaining factors constituting those of $\mathcal{M}_r$. The other original polynomial-time algorithm was given in [250]. Winkler's algorithm can be extended to the directed case, and this was shown by Walker [240].

There have been noticeable improvements to the naive implementation of Winkler's algorithm. One particular improvement is presented by Aurenhammer *et al* [7], which claims that the factorisation process is no more complex than sorting the graphs edges, and is at most $\mathcal{O}$(mlnn), where n is the number of vertices and m the number of edges. Whether this algorithm could be extended to deal with digraphs is uncertain.

The three algorithms by Feigenbaum-Hershberger-Schaffer, Winkler and Aurenhammer-Hagauer-Imrich are very different. The easiest to understand is Winkler's. Feder's algorithm has some similarities to the improved version of Winkler's algorithm, but you have to appreciate the theory of isometric embeddings that Graham, Winkler, and Feder have worked on to see the similarities. Winkler has an excellent survey paper on this theory in [251].

Of all the papers mentioned Winkler's survey is the most readable, and is a recommended starting point. However, it is not as readily available since it appeared as a book chapter and not as a standard journal article.

No polynomial algorithms that deal with labeled digraphs have been found, and one suspects that if they do exist, then they are unlikely to be polynomial. While the existence of such algorithms would certainly have a marked impact on the construction of $\mathcal{M}_r$, the digraphs algorithms provide a useful starting point, and as is discussed in the next section, $\mathcal{M}_r$ is still extractable in a reasonable time frame using these algorithms. From the above survey, [7] is used in the work described here and is detailed in appendix C.

Having formally defined the meaning of quotient, and surveyed the algorithms available to solve quotient problems, we now switch our attention to applying graph theory [93, 249, 195, 54] to solve the interface equation.

5.3 A graph theoretic approach

The first problem was to identify which graph product definition was closest, if not identical, to the parallel operator used within the interface equation. It transpires that the definition of parallel composition is identical with the Graph Cartesian product (GCP), under certain conditions. That the match isn't quite perfect led us to develop a number of novel techniques, and to the extension of CCS. By moving the problem to

this domain a number of characteristics appeared which seemed ripe for exploitation. Principle among these was *symmetry*. What can symmetry tell us about interacting machines? When solving the interface equation, the main problem is identifying those actions that occur due to internal communication between the two combined machines. The basic observation is that transitions which are intrinsic to the composite machine elements (that is the τ actions that ($\mathcal{M}_p$ and $\mathcal{M}_r$ do internally, without any communication) have a specific structure — that they obey certain symmetry constraints. On the other hand, actions which have arisen via internal communication between say $\mathcal{M}_p$ and $\mathcal{M}_r$, do not in general exhibit the symmetry expected. We first need a formal definition of the graph Cartesian product (GCP). From now on the term *graph* should be read as *labeled digraph* [54, 249] unless explicitly stated otherwise. The following definition is from [9].

5.3.1 Definition

Given two graphs G_1 and G_2 with $V(G_1) \cap V(G_2) = \emptyset$, the graph Cartesian product $G_1 \times G_2$ is the graph G_3 with vertex set $V(G_1) \times V(G_2)$ such that two vertices (u_1, u_2) and (v_1, v_2) of G_3 are adjacent if and only if either $u_1 = v_1$ and the (directed) edge $u_2v_2 \in E(G_2)$ or $u_2 = v_2$ and $u_1v_1 \in E(G_1)$.

G_3 then, consists of a set of ordered pairs as vertices, the components being the vertices of G_1 and G_2, connected by directed edges which are the edges of G_1 and G_2, and *only those edges*. Hence, given a directed edge $uv \in G_1$, which may be labeled by a parameter α say, then the labeled directed edge uv_α will be repeated in G_3 for each G_2 vertex. Likewise, each directed edge in G_2 will appear in G_3 for each vertex of G_1. This is the symmetry principle — an edge in either graph will be repeated in accordance with both the order and the structure of the other graph. This is a key

observation, since this only applies to edges which are composed via the GCP. Edges which are added after the two graphs, or form as a result of some transformation (e.g. topological homeomorphism or bisimulation based contraction) have been composed will not exhibit this property, and will destroy the inherent symmetry of the product graph. The symmetries associated with $G_1 \times G_2$ are dependent on the symmetries of G_1 and G_2, which in turn is dependent on the nature of $G_1 \cap G_2$, e.g. whether they are disjoint or have subgraphs in common. The symmetries associated with a group is intimately connected with its automorphism group Γ. A fundamental result from graph theory is that if G_1 and G_2 are relatively prime, then $\Gamma(G_1 \times G_2) \cong \Gamma(G_1) \otimes \Gamma(G_2)$, where '$\otimes$' is the group direct product. When the prime condition is not met the relationship is more complicated due to dependence effects. For a more detailed examination of graph symmetries and quotients see [9, 200, 149].

Now consider state machines, and in particular the state transition graphs associated with them. It is clear that the state transition graphs are no more than labeled digraphs. As such, we can form the product of the state transition graphs using the GCP. What do we find? Well it turns out that the GCP is identical, in terms of the resulting product structure, as one would have obtained if the machines were specified in CCS [1] and composed using Milners composition operator '|', with either no communication between $\mathcal{M}_p$ and $\mathcal{M}_r$ or with no communication actions hidden and the τ_c actions removed. By way of an example, consider the Milner's definition of the composition operator (see the Expansion Theorem [152]), where given $p \rightarrow^{\alpha} p'$ and $q \rightarrow^{\beta} q'$ then $p \mid q$ is defined as :-

$$\frac{p \rightarrow^{\alpha} p' \mid q \rightarrow^{\beta} q'}{p \mid q \rightarrow^{\alpha} p' \mid q,\ p' \mid q \rightarrow^{\beta} p' \mid q',\ p \mid q' \rightarrow^{\alpha} p' \mid q',\ p \mid q \rightarrow^{\beta} p \mid q'}$$

[1] CCS has proved an ideal formal language for this work. However, any other notation which has a natural graphical representation can be used.

Strictly speaking we do not need to write the denominator in full, but have done to illustrate the correspondence with the GCP. It is clear that there is some similarity here between the structure of the product graphs and the composition of the machines. From the point of view of category theory it appears that there may be a functor between the category of machines and composition to the category of graphs and product [2]. What is interesting is that via theorem 5.3.2 we are making topological, or structural observations about the machines. From this point of view the behavioural dynamics of the machines is of no consequence, and as such, issues such as nondeterminism, deadlock or divergence among others, can effectively be ignored. This poses an interesting set of questions concerning the relationship between machine structure and behaviour. This is discussed later in chapter 8. We now state the connection between GCP and CCS composition formally :-

5.3.2 Theorem

Given two arbitrary machines $\mathcal{M}_p$ and $\mathcal{M}_q$, such that $\forall\ \alpha \in \Lambda(p)\ \neg\exists\ \overline{\alpha} \in \Lambda(q)$, along with their state transition graph representations $\mathcal{G}_{\mathcal{M}_p}$ and $\mathcal{G}_{\mathcal{M}_q}$ respectively, then $\mathcal{G}_{\mathcal{M}_p|\mathcal{M}_q} \cong \mathcal{G}_{\mathcal{M}_p} \times \mathcal{G}_{\mathcal{M}_q}$.

Proof: We show that $G_{\mathcal{M}_p|\mathcal{M}_q}$ is isomorphic to $\mathcal{G}_{\mathcal{M}_p} \times \mathcal{G}_{\mathcal{M}_q}$. Firstly, an invertible mapping σ is defined where $\Xi : (p_i \mid q_i) \longmapsto (p_i, q_i)$. Let $(p_i \mid q_i) \rightarrow^{\alpha} (p_j \mid q_j)$ be a transition in $\mathcal{M}_p \mid \mathcal{M}_q$. The action α is intrinsic to either $\mathcal{M}_p$ or $\mathcal{M}_q$, hence either p_i does the transition and $q_i = q_j$ or q_i does the transition and $p_i = p_j$. But by the definition of the GCP and the mapping Ξ, either $q_i = q_j$ and $p_i p_j \in E(\mathcal{G}_{\mathcal{M}_p})$ or $p_i =$

[2] We will not pursue this here.

p_j and $q_iq_j \in E(\mathcal{G}_{\mathcal{M}_q})$. But p_ip_j and q_iq_j are merely the labeled directed edges from the state transition diagrams of the respective machines, and hence by definition $(p_i, q_i)(p_j, q_j)_\alpha \in E(\mathcal{G}_{\mathcal{M}_p} \times \mathcal{G}_{\mathcal{M}_q})$. Let $(p_i, q_i)(p_j, q_j)_\alpha \in E(\mathcal{G}_{\mathcal{M}_p} \times \mathcal{G}_{\mathcal{M}_q})$ be a directed labeled path between two vertices in $\mathcal{G}_{\mathcal{M}_p} \times \mathcal{G}_{\mathcal{M}_q}$. We need to show that there is a corresponding transition in $\mathcal{M}_p \mid \mathcal{M}_q$. Using $\Xi^{-1}(p_i, q_i) \longmapsto (p_i \mid q_i)$ and $(p_j, q_j) \longmapsto (p_j \mid q_j)$. By definition of the composition operator, there are two possibilities; either $(p_i \mid q_k) \rightarrow^\alpha (p_j \mid q_k)$ for some k, or $(p_k \mid q_i) \rightarrow^\alpha (p_k \mid q_j)$ for some k, in other words $\mathcal{M}_p$ does the transition and $\mathcal{M}_q$ remains fixed or $\mathcal{M}_q$ does the transition and $\mathcal{M}_p$ remains fixed. So for any labeled directed path in $\mathcal{G}_{\mathcal{M}_p} \times \mathcal{G}_{\mathcal{M}_q}$ we can find a corresponding transition in $\mathcal{M}_p \mid \mathcal{M}_q$. Hence, in terms of the underlying structure $G_{\mathcal{M}_p|\mathcal{M}_q} \cong \mathcal{G}_{\mathcal{M}_p} \times \mathcal{G}_{\mathcal{M}_q}$. QED. □

Two corollarys follow immediately from this theorem.

5.3.3 Corollary

$G_{(\mathcal{M}_p|\mathcal{M}_q)\backslash B} \cong G_{\mathcal{M}_p\backslash B} \times G_{\mathcal{M}_q\backslash B}$, where B is a set of actions not involved in communication.

Proof: Follows immediately from theorem 5.3.2. □

5.3.4 Corollary

In general $G_{(\mathcal{M}_p|\mathcal{M}_q)\backslash A} \ncong G_{\mathcal{M}_p\backslash A} \times G_{\mathcal{M}_q\backslash A}$ where A is a set of actions which contains only those actions which are involved in communication.

Proof: Follows immediately from theorem 5.3.2. □

Corollary 5.3.4 follows since communication between the respective machines can only occur via the '|' operator. In terms of the graphs, and the GCP, there is no implicit link between say an $\alpha \in \mathcal{M}_p$ and an $\overline{\alpha} \in \mathcal{M}_r$ – they are merely edge labels

and no communication takes place. The corollary highlights the distinction between behavioural (via '|') and structural (via GCP) views of a machine.

Corollary 5.3.3 is useful in that it implies that the algorithm to solve the equation can ignore non-communicating actions during quotient extraction. Hence, actions in $\mathcal{M}_p$ which are not involved in communication can be hidden since they provide no additional information in terms of generating $\mathcal{M}_r$ (clearly the same actions have to be hidden in $\mathcal{M}_q$ as well). The preceding theorem provides a formal link between the machine and graph domains. The symmetries induced by the GCP that are observed within the product graph will exist within the corresponding composed machine. The important point to observe here is that, as a result of this symmetry, it is possible to determine the embedded structure of the component graphs or machines. That is to say $\exists G_a$ and $G_b \subset G_{\mathcal{M}_p|\mathcal{M}_q} : G_a \cong \mathcal{G}_{\mathcal{M}_p}$ and $G_b \cong \mathcal{G}_{\mathcal{M}_q}$. What is more, there will be, due to symmetric nature of the GCP, a number of such graphs. The actual number will depend on the respective orders of the graphs concerned. If G_1 is of order m and G_2 of order n, then there will be m copies of G_2 and n copies of G_1 in $G_1 \times G_2$, though as will be seen later, some notions of equivalence, observational for instance, cause complications. This provides the lever for the research presented in this thesis, for if we can express the interface equation in terms of graphs, then we can use GCP symmetry and existing graph quotient algorithms to solve the equation. As a result of this we make the following conjecture.

5.3.5 Conjecture

Given $(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \approx \mathcal{M}_q$, then the problem of finding $\mathcal{M}_r \stackrel{S}{=} \frac{\mathcal{M}_q}{\mathcal{M}_p}$ can be transformed to finding $\mathcal{G}_{\mathcal{M}_r} \cong \frac{\mathcal{G}_{\mathcal{M}_q}}{\mathcal{G}_{\mathcal{M}_p}}$, if a solution exists, once the effect of A has been removed and where '$\stackrel{S}{=}$', structural equivalence, is defined in chapter 6.

The remainder of this chapter is devoted to demonstrating the validity of this conjecture. We shall refer to machines such as $\mathcal{M}_\tau$, which can be expressed as the quotient of other machines, as a *quotient machine.*

5.3.6 Internal communication

As was stated earlier, the occurrence of internal communication within the composed machines has a marked effect on the underlying structure of the state transition graphs. As a reminder, Milner defines internal communication as follows :-

$$\frac{p \rightarrow^{\alpha} p' \mid q \rightarrow^{\overline{\alpha}} q'}{p \mid q \rightarrow^{\alpha} p' \mid q,\ p' \mid q \rightarrow^{\overline{\alpha}} p' \mid q',\ p \mid q' \rightarrow^{\alpha} p' \mid q',\ p \mid q \rightarrow^{\overline{\alpha}} p \mid q',\ p \mid q \rightarrow^{\tau} p' \mid q'}$$

Thus the composition operator enforces action/dual-action [3] pairs to combine and form a τ action. As is apparent this is quite different to the structure we obtained before, the α and $\overline{\alpha}$ actions have combined to form the single τ action. Note that this τ transition does not possess the symmetry properties of the α and $\overline{\alpha}$ transitions, so given a structure such as this, we could in principle, identify the τ transition as not being intrinsic to $\mathcal{M}_p$ or $\mathcal{M}_q$, but arising from the communication process. Further, in this example, the τ action could be removed, leaving the machine with the structural properties expected when no communication occurs. Of course, it may be that the composed machines did a τ action prior to composition. But such actions, like any other, would still appear in the resulting machine in accordance with the symmetry principles expected from Def 5.3.1. It is the τ actions which result from inter-machine communication which do not. Clearly, it would be useful to distinguish between these τ-actions, this we do via the following definitions.

[3] Milner uses the term 'complementary' to describe these actions. The author feels that the term 'dual' is more appropriate and will hence use it throughout this thesis.

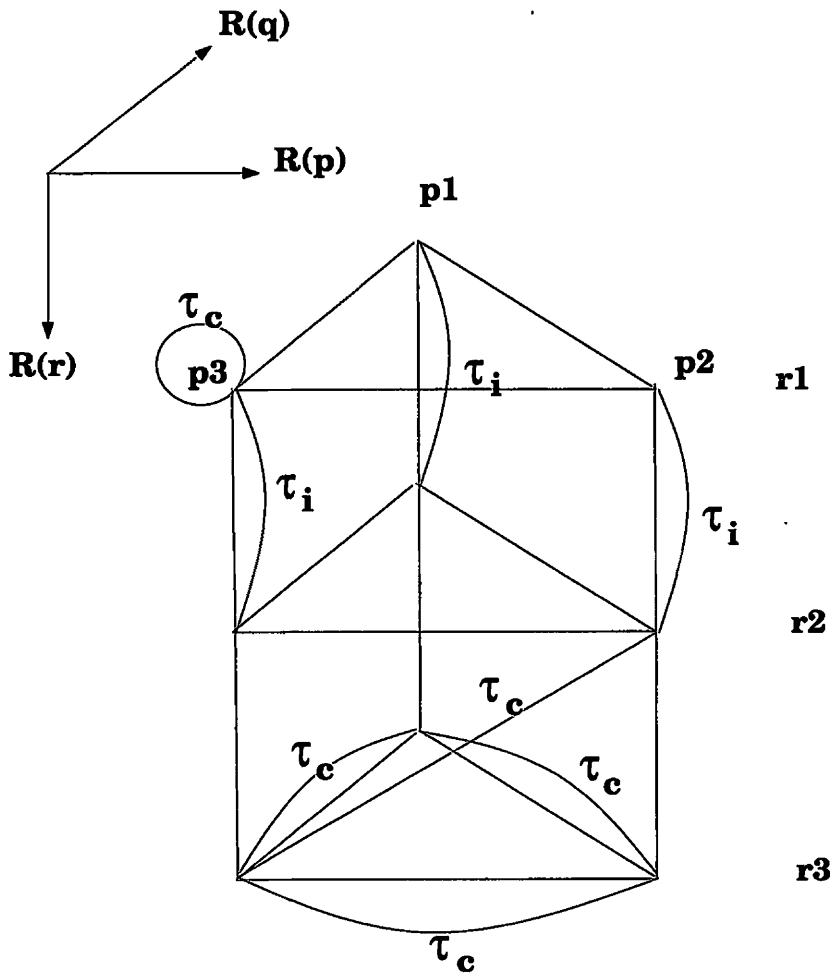


Figure 5.2: Intrinsic and communicating τ-actions. The intrinsic actions, τ_i, are τ actions performed by $\mathcal{M}_r$ or $\mathcal{M}_p$ without communication. The τ_c actions refer to actions which have arisen as a result of $\mathcal{M}_p$ and $\mathcal{M}_r$ communicating. This graph shows the possibilities that may emerge from a structural perspective. Intrinsic actions will also possess a degree of symmetry (due to the properties of '|' and the GCP. Communicating actions will typically be assymetric, but can be symmetric (see the transitions involving $(p_1, r_3), (p_2, r_3), (p_3, r_3)$). The distinction between symmetric τ_i and symmetric τ_c actions is the fact that the former involves no actions from $\mathcal{M}_p$, whilst the latter does. Actions that $\mathcal{M}_p$ does, and which are involved in communication, will not appear in $\mathcal{M}_q$.

5.3.7 Definition

Given an interface equation $(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \sim \mathcal{M}_q$, a τ action in $\mathcal{M}_q$ arising from a communication between the $\mathcal{M}_p$ and $\mathcal{M}_r$ machines will be referred to as a *communicating* τ and denoted τ_c. A τ action in $\mathcal{M}_q$ which exists in $\mathcal{M}_p$ or $\mathcal{M}_r$ will be called an *intrinsic* τ and denoted τ_i.

Referring to figure 5.2, we can see the different structural characteristics between the two forms of internal communication. One might argue that we could arrange $\mathcal{M}_p$ and $\mathcal{M}_r$ to communicate in such a way as to mimic the symmetry of the intrinsic τ-actions. Well, indeed we can, but only via the synchronisation of actions from the two machines. Since we know $\mathcal{M}_p$, and what actions are involved in communication, then these apparently 'symmetric' τ_c-actions can still be distinguished from the intrinsic τ_i's (figure 5.2).

This demonstrates, albeit very simply, the basic idea behind the algorithm presented in this thesis — identify which τ actions were generated by communication (the τ_c-actions); remove them in some way; return the action/dual-action pairs that formed them and as a consequence generate the structure expected from the composition process where no communication has occurred; and finally apply the quotient algorithm. In terms of the examples given so far, this is relatively straightforward. However, in fact there are a number of complexities which need to be considered.

5.3.8 Hiding

In the previous section we saw how Milners composition operator, as defined in [152], enforced action/dual-action pairs to communicate. The actions themselves do not disappear — the machine can still perform them provided they are not members

of A. What has happened is that for certain transitions, events synchronise and communicate in a way that is not observable to the outside world. Practitioners using CCS though, seem to have adopted a convention that the actions that appear in Milners restriction set (which we have termed generically A), are precisely those, and only those, that are involved in communication. In this case the inference equation looks quite different :-

$$\frac{p \rightarrow^{\alpha} p' \mid q \rightarrow^{\overline{\alpha}} q'}{(p \mid q)\backslash\{\alpha\} \rightarrow^{\tau_c} (p' \mid q')\backslash\{\alpha\}\ (p' \mid q)\backslash\{\alpha\}(p \mid q')\backslash\{\alpha\}}$$

Strictly speaking however, the restriction set need not be confined to, or even include actions which are involved in communication. This notion of hiding seems counter-intuitive, since what the interface engineer really requires is a mechanism whereby he or she can specify which actions must communicate. Further, this notion of communication would seem to suggest that allowing the resulting machine to still carry out the action/dual-action separately has little practical use (though it clearly has considerable theoretical significance). We should like to point out that misgivings concerning Milner's composition and hiding operator are not confined to ourselves, see for example [29]. As a result of this we have defined new hiding and composition operators the details of which can be found in chapter 6. For this chapter we stick to Milner's notation, however, the reader should be aware, as pointed out in the introduction, that our use of it will not be strict as per Milner's definitions. The object of this approach is to demonstrate clearly the problems that arise when using Milner's notation in a graph theoretic environment, and hence the need for an extended CCS syntax and semantics. In the remainder of this chapter Milner's restriction set A will be taken to contain only communicating actions and no others. Thus we use the following rule :-

$$\frac{p \rightarrow^{\alpha} p' \mid q \rightarrow^{\overline{\alpha}} q'}{p \mid q \rightarrow^{\alpha} p' \mid q,\ p' \mid q \rightarrow^{\overline{\alpha}} p' \mid q',\ p \mid q' \rightarrow^{\alpha} p' \mid q',\ p \mid q \rightarrow^{\overline{\alpha}} p \mid q',\ p \mid q \rightarrow^{\tau} p' \mid q'}$$

So the problem can be viewed thus — if a transformation can be found which effectively removes all the communicating τ-actions and simultaneously replaces the actions lost due to synchronisation, then the graph quotient algorithms can be applied directly. This chapter discusses such a transformation, τ-splitting. The transformation is concerned with the actions that matter — the τ_c-actions — and effectively ignores all others. The transformation rests on one key characteristic inherent within product graphs — namely symmetry. The next section describes τ-splitting in detail.

5.3.9 Tau splitting

This section will provide the necessary syntax and semantics of τ-splitting, beginning with formal definitions, and closing with an example. This splitting process is key to the application of the graph quotient algorithms, since it restores the symmetries of the graph product form (remember the *graphs* are the state transition diagrams of machines). Before dealing with the formal definitions, we will use a simple example to highlight the issues involved.

The aim of the τ-splitting transformation is to resolve a τ_c action into its component actions — that is the actions which combined to form it. If $p_1 \rightarrow^{\alpha} p_2$ and $r_1 \rightarrow^{\overline{\alpha}} r_2$ are composed using parallel composition we get $p_1 \mid r_1 \rightarrow^{\tau_c} p_2 \mid r_2$ (assuming the original α and $\overline{\alpha}$ actions are hidden). The τ-splitting transformation, which is denoted by Υ, takes as its argument the given transition and 'splits' the τ_c in order to recover the α and $\overline{\alpha}$ transitions. Hence, the τ_c is removed from the machine and a number of new transitions added.

K-sets and Π-planes

Given an interface equation, the inference is that the $\mathcal{M}_q$-machine is in fact the composition of two other machines. The plan of action is to remove communication from the $\mathcal{M}_q$ machine using τ-splitting, and then to derive the quotient of $\mathcal{M}_q$ and $\mathcal{M}_p$, which will be the interfacing machine $\mathcal{M}_r$ – providing a solution exists. Hence we are given one component, the $\mathcal{M}_p$-machine, but need to find an $\mathcal{M}_r$-machine such that $\mathcal{M}_p \mid \mathcal{M}_r$ is equivalent in some way to $\mathcal{M}_q$. The notion of equivalence is an important issue, and one which will be covered in more depth later, but for now assume that by equivalence we mean isomorphism. Now if $\mathcal{M}_q$ is the composition of two machines then each state of $\mathcal{M}_q$, q_1 say, is in fact an ordered pair (p_1, r_1). The first step is to partition the $\mathcal{M}_q$ machine into distinct sets via special sets called $K(r)$-sets. These $K(r)$-sets are similar to those defined by Shields and Martin, details of which can be found in chapter 4. K-sets were originally defined by Shields and are fundamental to the discarding and constructive algorithms. Their usefulness here is that the internal structure of K-sets is intimately related to the structure of $\mathcal{M}_p$. In addition, each K-set is associated with one, and only one, state of $\mathcal{M}_r$. The principle difference is that their definition allowed a distinct q to appear in more than one $K(r)$-set, whereas our approach requires uniqueness. Since we are constructing a formal partition of the $\mathcal{M}_q$-machine, the $K(r)$-sets by themselves are not sufficient. Structural equivalence requires that these $K(r)$-sets be projected onto $\mathcal{M}_p$, as the $K(r)$-sets themselves do not form a partition of $\mathcal{M}_q$. Hence each state q in $\mathcal{M}_q$ is mapped onto one, and only one, p-state in $\mathcal{M}_p$ (see definition 5.3.11). It is important to make this distinction between $K(r)$-sets and the partitions generated via the projection mechanism, which arises from the equivalence notion used (they used observational equivalence). We will refer to the partition planes as $\Pi(r)$-planes. $\Pi(r)$-planes are defined as follows :-

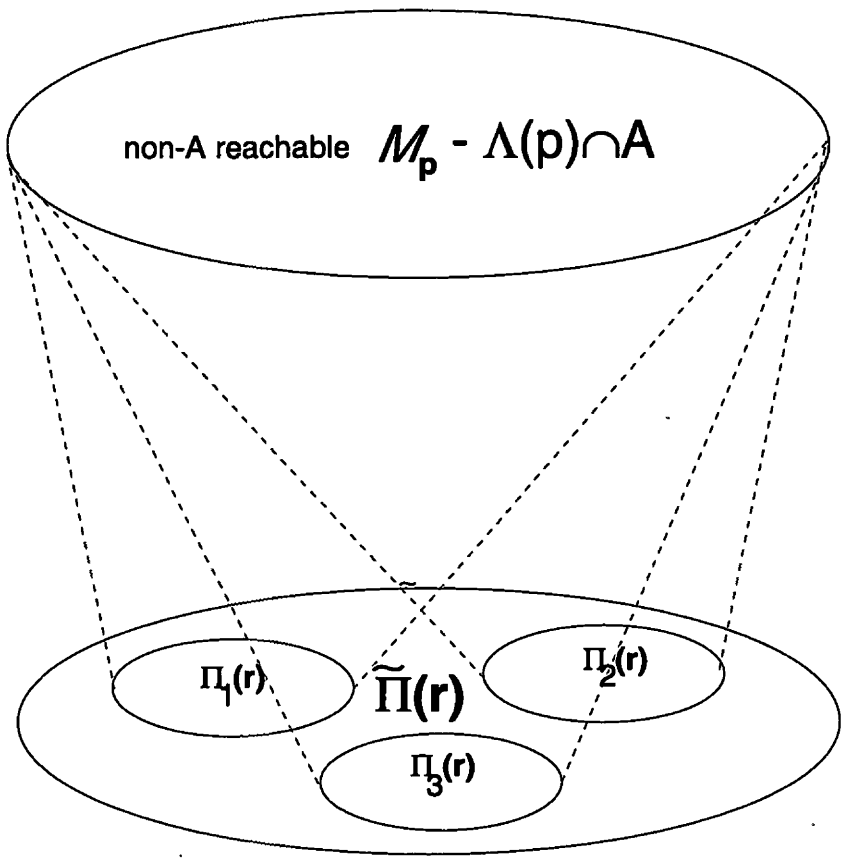


Figure 5.3: The Π and $\tilde{\Pi}$ planes

5.3.10 Definition

Given a $K(r)$-set generated by Martin's algorithm [140], the associated $\Pi(r)$-plane is defined as the partition component derived from $\sigma : K(r) \rightarrow \mathcal{M}_p - A \cup \{r\}$ (see definition 5.3.11). Note each $\Pi(r)$-plane $\subseteq \mathcal{M}_p$. Hence there is an injective mapping between $K(r)$-sets and $\Pi(r)$-planes.

Let us be clear about what is going on here. The K-sets provide a useful means of assigning each state q in $\mathcal{M}_q$ to its associated state in $\mathcal{M}_p$. Now a K-set consists of ordered pairs (p_i, q_i), which assigns the state q_i to p_i. Since the r state is implicit we can infer that the state q_i corresponds to $p_i \mid r$. Since we require more structure

than just ordered pairs, we use the σ Def 5.3.11 mapping to identify which subgraph of $\mathcal{G}_{\mathcal{M}_p}$ the K-set corresponds to. It is this subgraph which we refer to as the Π-plane. Note also the secondary value of the K-sets, since under observational and strong equivalence a q state may appear in more than one K-set, $K(r_1)$ and $K(r_2)$ say. This is a useful tool to identify where machine reduction has taken place since under structural equivalence each q must be in one and only one K-set.

The affect of the partitioning is to divide the $\mathcal{M}_q$ machine into a number of planes. These planes are connected by actions from the $\mathcal{M}_r$-machine and τ_c actions, and no others. All the $\mathcal{M}_p$-machine transitions not involved in communication lie within the $K(r)$-sets. This is the central observation regarding τ-splitting, since the τ_c transitions between $K(r)$-sets arose from communication between the $\mathcal{M}_p$ and $\mathcal{M}_r$ machine. If the communicating actions are removed, and the 'lost' actions which combined to form it replaced we would find that the transitions within the $\Pi(r)$-planes are all the transitions performed by the $\mathcal{M}_p$-machine, and the transitions between $\Pi(r)$-planes are the transitions performed by the $\mathcal{M}_r$-machine. The resulting structure is now in a form whereby the quotient algorithms can be used. Before we can formally define and describe the mechanics of Υ a number of preliminary definitions needs to be made.

The σ mapping

5.3.11 Definition

The mapping σ is defined as follows :-

$$\begin{aligned} \sigma &: R(q) \longrightarrow R(q) \times R(p) \times R(r) \\ \sigma &: \{q \longmapsto (q,p,r) \mid p \mid r \stackrel{S}{=} q\} \end{aligned}$$

Hence, for each state q of $\mathcal{M}_q$, we can assign a unique triple which satisfies the condition that $p \mid r$ is equivalent to q, where the equivalence is that incorporated

within the interface equation. This mapping is unique as a consequence of structural equivalence. It is interesting to note that both the discarding and constructive algorithms also require this mapping. The method to construct it is given in [217, 218], but in these cases the uniqueness property is lost. In $R(p) \times R(q) \times R(r)$ space, these components lie in the $R(p) \times R(q)$ plane, being indexed via the unique state from $\mathcal{M}_r$. The internal actions of each $\Pi(r)$-plane either come from $\Lambda(p)$ or are τ_c actions arising from communication between $\mathcal{M}_p$ and $\mathcal{M}_r$. Further, actions between $\Pi(r)$-planes must be either τ_c-actions or come from $\mathcal{M}_r$. One can immediately see the closeness of this structure to that expected from the GCP by looking at figure 5.4.

Reachability

The K-set as defined in chapter 4 requires a condition on $\mathcal{M}_p$ which is to restrictive for our purposes — and that is what is referred to as non-A/B p-reachability. This means that each state in K can reach, via a directed path, any other state in K via actions not in A or B. In the semantics of graph theory this amounts to saying that the state transition graph of a machine must be strongly connected [9] after hidden actions have been removed. Since graph quotient algorithms can work with weakly connected graphs we have defined a new $\Pi(r)$-plane, called the $\tilde{\Pi}$-plane which is the union of $\Pi(r)$-planes. While each $\Pi(r)$-plane is connected via directed paths, each $\tilde{\Pi}$-plane need not be. In the $\tilde{\Pi}$-planes the states are connected via undirected paths. The resultant partition of $\mathcal{M}_q$ is not necessarily strongly connected; our condition on the $\tilde{\Pi}$-plane is that it is weakly connected.

5.3.12 Definition

$\tilde{\Pi}(r) = \bigcup_i \Pi_i(r)$ for some r and i, where $\Pi_i(r)$ is the set of $\Pi(r)$-planes associated with a given r, and where i indicates that the union is taken over the connected

(subgraphs) of $\mathcal{M}_p$ in $\mathcal{M}_q$ for each r.

But we could go further and actually have a disconnected $\tilde{\Pi}$-plane. Although quotient graph algorithms cannot deal with disconnected graphs in general, this need not be a problem. There are two options in terms of addressing disconnected $\tilde{\Pi}$ sets. The first uses a technique proposed by Kristol et al [63], where dummy actions (edges) are added. While the context of their work is quite different, it could in principle be applied here. The second concerns the nature of the GCP. Assume that $\mathcal{M}_p$ is the disconnected union of a number of machines $\mathcal{M}_p^i$, and that it is composed with another $\mathcal{M}_r$. Now

$$\begin{aligned}
\mathcal{M}_q &= \mathcal{M}_p \mid \mathcal{M}_r \\
&= \bigcup_i \mathcal{M}_p^i \mid \mathcal{M}_r \\
&= (\mathcal{M}_p^1 \mid \mathcal{M}_r) \cup (\mathcal{M}_p^2 \mid \mathcal{M}_r) \cup \cdots \cup (\mathcal{M}_p^i \mid \mathcal{M}_r) \\
&\Rightarrow \mathcal{M}_r \sim \frac{\mathcal{M}_q}{\mathcal{M}_p^i} \text{ for any } i.
\end{aligned}$$

While the initial processing would need to consider all the $\mathcal{M}_p$ components, the quotient algorithm would need to consider only one, since each component is composed with a complete $\mathcal{M}_r$. Of course the assumption here is that $\mathcal{M}_r$ is connected. If it isn't the situation is more complicated, but the author is convinced that a method could be derived to address even this problem (see chapter 8).

The derivation of these planes could be achieved using the techniques used in the constructive and discarding algorithms with only minor modifications. The additional steps being the formation of the $\Pi(r)$-planes via symmetric closure (section 5.4), and the subsequent construction of the $\tilde{\Pi}$-planes. The approach would be to examine every state in $\mathcal{M}_p$ to acquire the complete set of $\Pi(r)$-planes associated with a particular r.

We have now established a method of identifying the sub-graphs of $\mathcal{G}_{\mathcal{M}_q}$ which arise from $\mathcal{G}_{\mathcal{M}_p} \times \mathcal{G}_{\mathcal{M}_r}$. These correspond to the Π-planes. Before returning to τ-splitting we need one more definition which does for $\mathcal{M}_r$ what the K-sets do for $\mathcal{M}_p$.

5.3.13 Definition

$L(p) \subset \mathcal{M}_q$ is defined as the set of ordered pairs (p, r_i) and associated action set $\Lambda((p, r_i))$, where $p \in R(p), r_i \in R(r)$ and $\Lambda((p, r_i)) = \Lambda(q) - \Lambda(p) \cup \{\tau_c\}$.

Whereas the K and $\tilde{\Pi}$ sets were in the $R(p) \times R(q)$-plane and indexed by a state in $\mathcal{M}_r$, the $L(p)$-sets are in the $R(p) \times R(r)$-plane and indexed by a state in $\mathcal{M}_p$.

Tau splitting

5.3.14 Definition

The τ-splitting relation Υ defines the following mapping :-

$$\Upsilon \;:\; R(q) \times \{\tau_c\} \times R(q) \longrightarrow R(q) \times \Lambda(p) \times R(q) \times R(q) \times \bar{\Lambda}(p) \times R(q)$$

$$\Upsilon \;:\; (q_i, \tau_c, q_j) \longmapsto \{< (q_i, \alpha, q_k)_{\dagger r} \mid (q_i, \overline{\alpha}, q_l)_{\dagger p} >: (p, r_i) \rightarrow^{\alpha} (p', r_i) \in \tilde{\Pi}(r_i)\ \forall\ r_i \in R(r) \text{ and } (p_i, r) \rightarrow^{\overline{\alpha}} (p_i, r) \in L(p_i)\ \forall\ p_i \in R(p)\}$$

Given an interface equation $(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \sim \mathcal{M}_q$. Assume there exists a transition $p \rightarrow^{\alpha} p'$ in $\mathcal{M}_p$, where $\alpha \in A$. By our definition of the hiding operator, all actions in the communication set A will be hidden in $\mathcal{M}_q$. However, there will be τ_c actions in $\mathcal{M}_q$ which have arisen via the actions in A. Stated more formally, given $p \rightarrow^{\alpha} p'$ in $\mathcal{M}_p$ with $\alpha \in A$, then $\neg\exists q_i, q_j : q_i \rightarrow^{\alpha} q_j$ in $\mathcal{M}_q$. But by definition 5.3.11 $\exists q_k, q_l \in R(q), r_m, r_n \in R(r) : q_k \rightarrow^{\tau_c} q_l$ with $q_k \sim (p \mid r_m)\backslash A$ and $q_l \sim (p' \mid r_n)\backslash A : (p \mid r_m)\backslash A \rightarrow^{\tau_c} (p' \mid r_n)\backslash A$. Since $p \rightarrow^{\alpha} p'$, it follows that $r_m \rightarrow^{\overline{\alpha}} r_n$. But by definition 4.4.5 and definition 5.3.11, each r is associated with a $\tilde{\Pi}$-set, hence

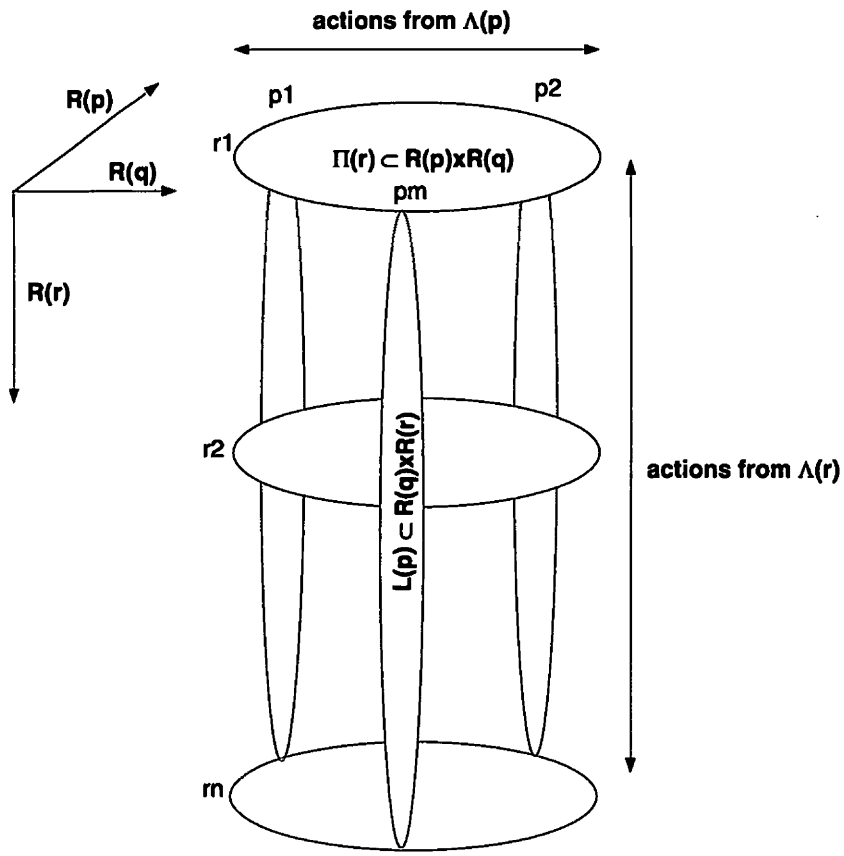


Figure 5.4: The $\tilde{\Pi}$ and L sets. The L-sets are the $\mathcal{M}_r$ equivalent of the $\tilde{\Pi}$-sets. Like the $\tilde{\Pi}$-sets, which are formed to deal with disconnected $\mathcal{M}_p$ machines, the L-sets are intended to address potential $\mathcal{M}_r$ disconnectedness.

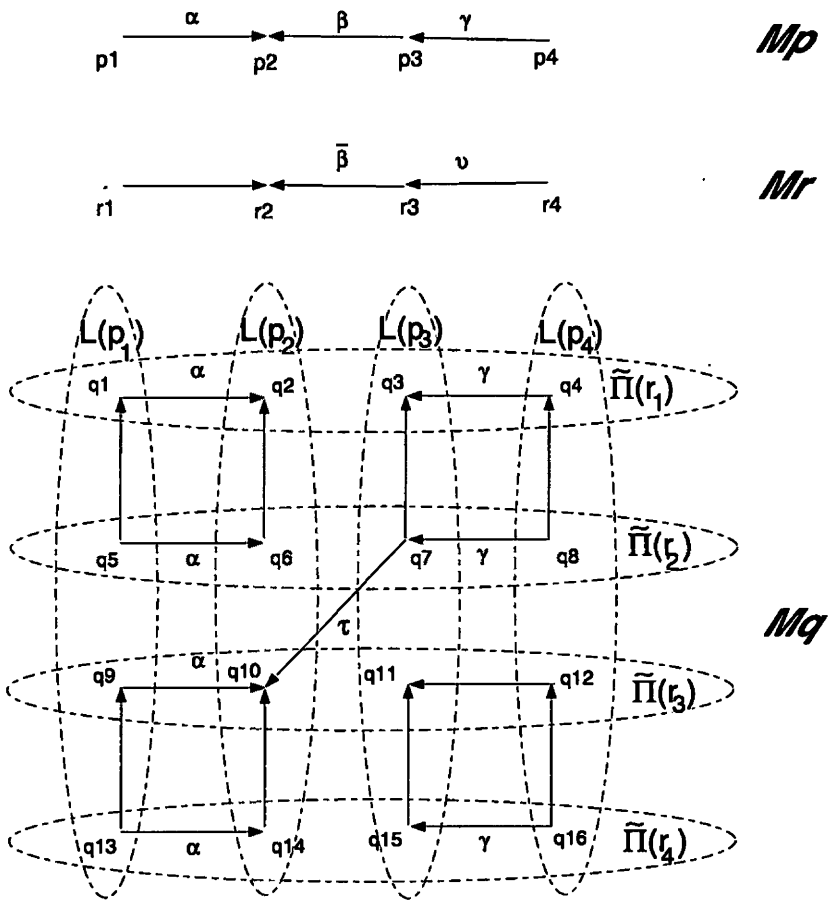


Figure 5.5: Machine $\mathcal{M}_q$

$\tilde{\Pi}(r_m) \rightarrow^{\overline{\alpha}} \tilde{\Pi}(r_n)\ \forall\ p \in \mathcal{M}_p$ (note that it may be the case that $r_m = r_n$). By the definition of the GCP and Theorem 5.3.2 $L(p) \rightarrow^{\alpha} L(p')\ \forall\ r \in \mathcal{M}_r$.

In terms of Υ, $q_k \rightarrow^{\tau_c} q_l$ is split and *projected* onto the underlying $\mathcal{M}_p$ and $\mathcal{M}_r$ components as the 6-tuple $< (q_i, \alpha, q_k)_{\dagger r} \mid (q_i, \overline{\alpha}, q_l)_{\dagger p} >$, where '†' indicates that the action is repeated in the plane indicated.

The splitting process is best illustrated with a diagram. Figure 5.5 shows the machine $\mathcal{M}_q$ which arises from the the composition of the following machines via the interface equation $(\mathcal{M}_p \mid \mathcal{M}_r)\backslash\{\beta\} \sim \mathcal{M}_q$:-

$$
\begin{aligned}
p_1 &\Leftarrow \alpha p_2 \\
p_2 &\Leftarrow \beta p_3 \\
p_3 &\Leftarrow 0 \\
p_4 &\Leftarrow \gamma p_3
\end{aligned}
$$

$$
\begin{aligned}
r_1 &\Leftarrow 0 \\
r_2 &\Leftarrow \mu r_1 + \overline{\beta} r_3 \\
r_3 &\Leftarrow 0 \\
r_4 &\Leftarrow \nu r_3
\end{aligned}
$$

Composing $\mathcal{M}_p$ and $\mathcal{M}_r$ gives :-

$$
\begin{aligned}
q_1 &\Leftarrow \alpha q_2 \\
q_2 &\Leftarrow 0 \\
q_3 &\Leftarrow 0 \\
q_4 &\Leftarrow \gamma q_3 \\
q_5 &\Leftarrow \mu q_1 + \alpha q_6 \\
q_6 &\Leftarrow \mu q_2 + \tau_c q_{11} \\
q_7 &\Leftarrow \mu q_3 \\
q_8 &\Leftarrow \mu q_4 + \gamma q_7 \\
q_9 &\Leftarrow \alpha q_{10} \\
q_{10} &\Leftarrow 0 \\
q_{11} &\Leftarrow 0 \\
q_{12} &\Leftarrow \gamma q_{11}
\end{aligned}
$$

$$q_{13} \Leftarrow \nu q_9 + \alpha q_{14}$$

$$q_{14} \Leftarrow \nu q_{10}$$

$$q_{15} \Leftarrow \nu q_{11}$$

$$q_{16} \Leftarrow \nu q_{12} + \gamma q_{15}$$

We will now assume that $\mathcal{M}_r$ is unknown, and demonstrate how τ-splitting operates given $\mathcal{M}_p$ and $\mathcal{M}_q$. Notice that a number of states are deadlocked. Previous algorithms, with the exception of the discarding algorithm and the constructive algorithm with merge-split, require the machines to be deadlock-free. Since the approach here is a structural one, such issues are of no consequence and have no impact on the algorithm.

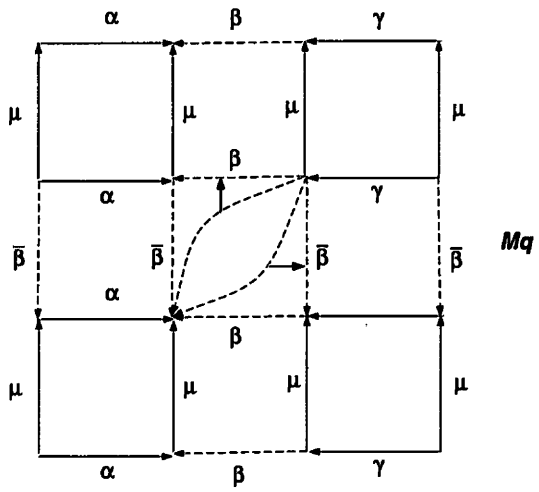


Figure 5.6: The projection of the split τ onto the $\mathcal{M}_p$ and $\mathcal{M}_r$ machines

Returning to the diagram, each state q has been assigned to a triple using σ. Each $\tilde{\Pi}$-plane is identified, and associated with a single state of $\mathcal{M}_r$ using the symmetry rules. Notice that under structural equivalence each $\tilde{\Pi}$-plane is identical with respect to internal transitions, and that the only differences are due to the transitions between $\tilde{\Pi}$-planes. this is not true in general for say strong and observational equivalence.

Application of the τ-splitting operator does two things :-

1. It removes the τ_c action from $\mathcal{M}_q$.

2. It projects the $\beta, \overline{\beta}$ pair onto the underlying machines

This is shown graphically in figures 5.6 and 5.7. The $q_7 \rightarrow^{\tau} q_{10}$ in figure 5.5 is the action which is split. Figure 5.6 shows the first stage of the splitting process — namely identifying the appropriate action/dual-action pair and projecting these onto the required q-states by first fixing the underlying p component, and then the r. The second stage (figure 5.7) shows the resulting actions now mapping the relevant $\tilde{\Pi}$ and L planes.

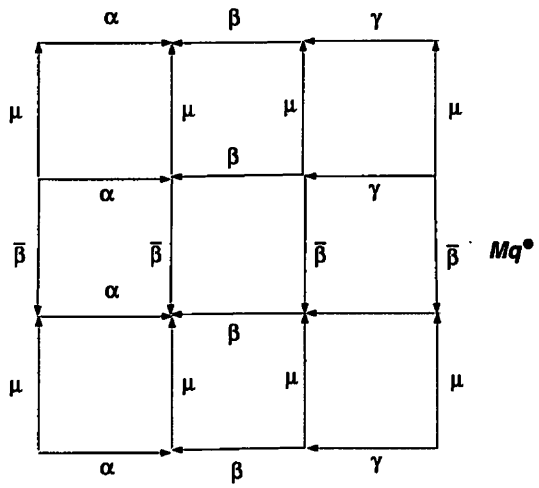


Figure 5.7: $\mathcal{M}_q^{\bullet}$ – ready for quotient extraction

The resulting state transition graph has the required structure enabling us to apply the quotient algorithm. An $\mathcal{M}_q$ which has all τ_c actions split will be denoted $\mathcal{M}_q^{\bullet}$. The following result follows from Theorem 5.3.2 and Def 5.3.3.

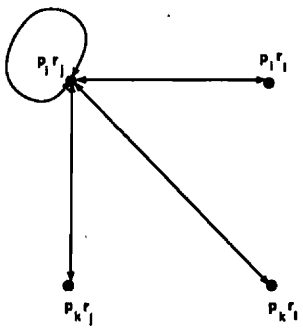


Figure 5.8: Possible τ derivations in $\mathcal{M}_q$

5.3.15 Lemma

Let $\mathcal{M}_q \stackrel{S}{=} (\mathcal{M}_p \mid \mathcal{M}_r)\backslash(A \cup B)$ then $\mathcal{M}_q^{\bullet} \stackrel{S}{=} (\mathcal{M}_p \mid \mathcal{M}_r)\backslash B$ and $G_{\mathcal{M}_q^{\bullet}} \cong G_{(\mathcal{M}_p|\mathcal{M}_r)\backslash B} \cong G_{(\mathcal{M}_p\backslash B|\mathcal{M}_r\backslash B)}$.

Proof: Follows from theorem 5.3.2, corollary 5.3.4 and corollary 5.3.3. □

5.3.16 Possible τ configurations

The previous section identified the communicating τ actions as an obstacle in applying quotient graph techniques to find a solution to the interface equation. It follows therefore, that any algorithm that is derived to transform the $\mathcal{M}_q$ machine need only look at these particular τ actions — it can ignore all the other actions during the splitting process. The key problem is to distinguish between true communicating τ actions, and those which are intrinsic to either the $\mathcal{M}_p$ or $\mathcal{M}_r$ machines. It is here that the symmetry properties inherent with the GCP become particularly useful — though as will be seen, there are certain pathological examples which pose interesting questions. There are four basic cases to consider, which are illustrated in figure 5.8 illustrates the possible derivations.

1 $\mathcal{M}_p$ and $\mathcal{M}_r$ change state $(p_i, r_j) \rightarrow^{\tau} (p_k, r_l)$

2 $\mathcal{M}_p$ state change, $\mathcal{M}_r$ fixed $(p_i, r_j) \rightarrow^{\tau} (p_k, r_j)$

3 $\mathcal{M}_p$ fixed, $\mathcal{M}_r$ state change $(p_i, r_j) \rightarrow^{\tau} (p_i, r_l)$

4 $\mathcal{M}_p$ and $\mathcal{M}_r$ fixed (recursive) $(p_i, r_j) \rightarrow^{\tau} (p_i, r_j)$

Each of these cases shall now be dealt with in turn.

Case 1

Case 1 represents a true communicating event, and hence can be immediately marked as a τ_c. Cases 2, 3 and 4 are more complicated since whether or not they are τ_c actions depends on other factors.

Case 2

Here a part of the $\mathcal{M}_q$ machine does a τ action, which in terms of the underlying components ($\mathcal{M}_p$ and $\mathcal{M}_r$) results in a $\mathcal{M}_p$ state change, with $\mathcal{M}_r$ remaining fixed. The first step is to look at the difference between $\mathcal{M}_p$ and $\tilde{\Pi}$. Denote this by $\Delta(\mathcal{M}_p, \tilde{\Pi})$, which will consist of two sets, the first being the set of states $\mathcal{M}_p$ contains but $\tilde{\Pi}$ doesn't, the second the corresponding set of actions. Hence the interest is in those actions that $\mathcal{M}_p$ possessed, but which $\tilde{\Pi}$ no longer does. The inference is that these missing actions have communicated with the dual actions in $\mathcal{M}_r$, creating new τ actions.

Another observation concerns the symmetry of G_q. Due to the nature of the cartesian product, any $\mathcal{M}_p$-intrinsic actions are duplicated in the $\mathcal{M}_r$-plane. That is, if $p_i \rightarrow^{\tau} p_j$ then $(p_i, r) \rightarrow^{\tau} (p_j, r) \; \forall \; r$. Such τ actions shall be designated τ_i. The same applies for τ_i actions that the $\mathcal{M}_r$ machine carries out, that is $(p, r_i) \rightarrow^{\tau} (p, r_j) \; \forall \; p$. However, consider the following CCS equations :-

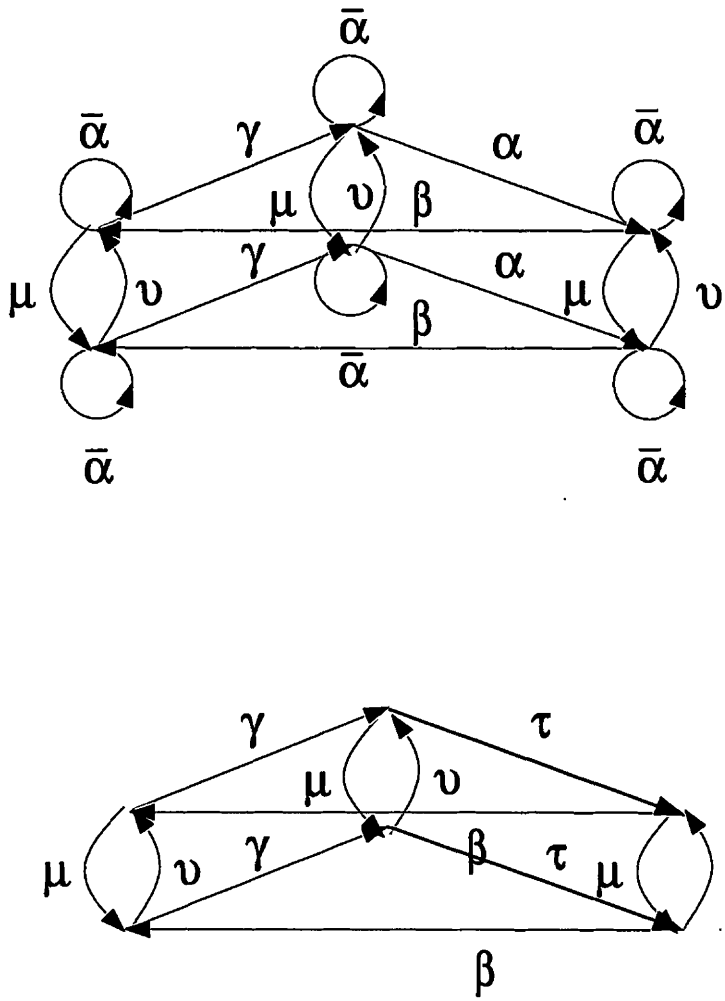


Figure 5.9: Possible τ derivations in $\mathcal{M}_q$ with $(p_i, r_j) \rightarrow^{\tau} (p_k, r_j)$

$$
\begin{aligned}
p_1 &\Leftarrow \alpha p_2 \\
p_2 &\Leftarrow \beta p_3 \\
p_3 &\Leftarrow \gamma p_1
\end{aligned}
$$

$$
\begin{aligned}
r_1 &\Leftarrow \overline{\alpha} r_1 + \mu r_2 \\
r_2 &\Leftarrow \overline{\alpha} r_2 + \nu r_1
\end{aligned}
$$

The resulting $\mathcal{M}_q$ machine exhibits the structure shown in figure 5.9. The point to observe is that whereas with τ_i actions $\Delta(\mathcal{M}_p, \tilde{\Pi})$ may be empty, in the example above this certainly won't be the case, since communication has clearly taken place, and the $\mathcal{M}_p$ machine has lost it's α actions.

Case 3

Here a part of the $\mathcal{M}_q$ machine does a τ action, which in terms of the underlying components ($\mathcal{M}_p$ and $\mathcal{M}_r$) results in a $\mathcal{M}_r$ state change, with $\mathcal{M}_p$ remaining fixed.

$$
\begin{aligned}
p_1 &\Leftarrow \alpha p_1 + \beta p_2 \\
p_2 &\Leftarrow \alpha p_2 + \gamma p_3 \\
p_3 &\Leftarrow \alpha p_3 + \delta p_1
\end{aligned}
$$

$$
\begin{aligned}
r_1 &\Leftarrow \overline{\alpha} r_2 \\
r_2 &\Leftarrow \epsilon r_1
\end{aligned}
$$

The resulting $\mathcal{M}_q$ machine exhibits the structure shown in figure 5.10. With no communication this has the same structure as the Case#2 example. However, the communication transforms the graph into a quite different structure. Again, the observation is that $\mathcal{M}_p$ and $\tilde{\Pi}$ contain all the information necessary to transform the

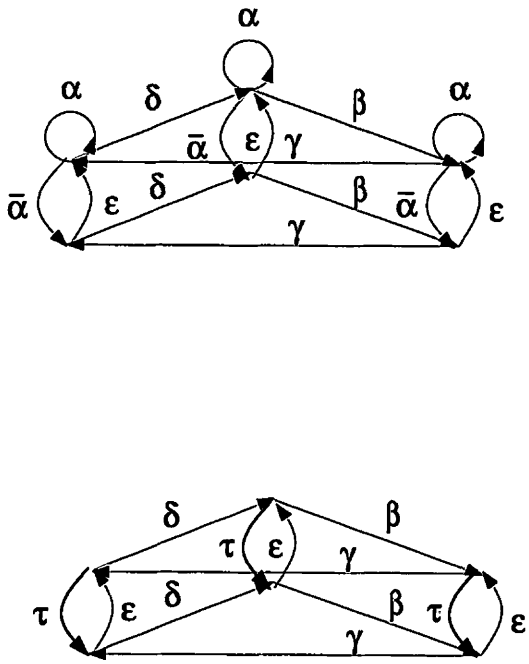


Figure 5.10: Possible τ derivations in $\mathcal{M}_q$ with $(p_i, r_j) \rightarrow^{\tau} (p_i, r_l)$

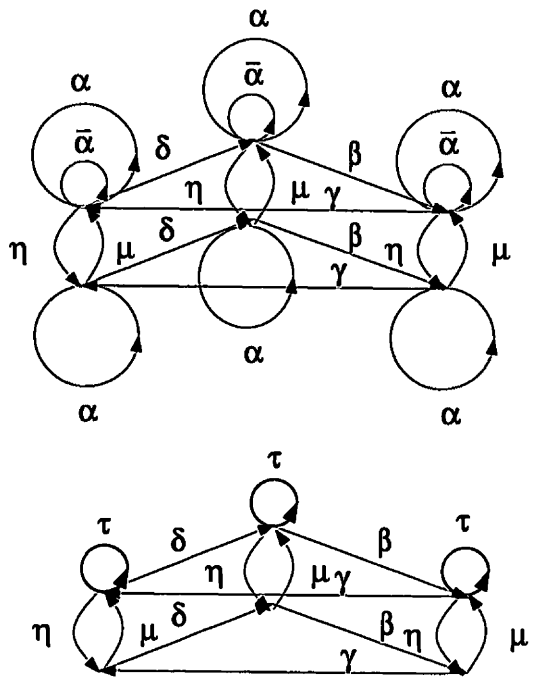


Figure 5.11: Possible τ derivations in $\mathcal{M}_q$ with $(p_i, r_j) \rightarrow^{\tau} (p_i, r_j)$

$\mathcal{M}_q$ machine into the appropriate structure, which amounts to reversing the communication mechanism.

Case 4

Here a part of the $\mathcal{M}_q$ machine does a τ action, which in terms of the underlying components ($\mathcal{M}_p$ and $\mathcal{M}_r$) results in both $\mathcal{M}_p$ and $\mathcal{M}_r$ remaining fixed. The options here are that $\mathcal{M}_p$ carries out a recursive τ-action with $\mathcal{M}_r$ remaining fixed, $\mathcal{M}_r$ does a recursive τ-action, or $\mathcal{M}_r$ and $\mathcal{M}_p$ communicate via recursive dual actions. Such a machine is specified below and shown in figure 5.11.

$$
\begin{aligned}
p_1 &\Leftarrow \alpha p_1 + \beta p_2 \\
p_2 &\Leftarrow \alpha p_2 + \gamma p_3 \\
p_3 &\Leftarrow \alpha p_3 + \delta p_1
\end{aligned}
$$

$$
\begin{aligned}
r_1 &\Leftarrow \overline{\alpha} r_1 + \mu r_2 \\
r_2 &\Leftarrow \nu r_1
\end{aligned}
$$

In terms of identifying these, the process is as before, look at $\tilde{\Pi}$ and identify whether any missing actions could be responsible for the $\mathcal{M}_q$ structure observed.

This concludes the formal description of τ-splitting and the associated theory. Before proceeding to the algorithm and its validation, we introduce the notion of symmetric closure.

5.4 Symmetric closure

There are a number of alternative notions of equivalence within CCS (and other languages). Of particular interest here is strong and observational equivalence. These notions of equivalence are useful in terms of identifying observationally degenerate states within a machine, which can then be collapsed thus reducing the number of states. Clearly, this is a useful property since by reducing the number of states, analytical processing (where the processing time is typically a function of the number of states) effort can be reduced. However, from the structural perspective this machine reduction can present problems.

It will be clear by now that the theory described in this thesis depends heavily upon the notion of symmetry and the GCP. Machine reduction has an adverse affect on this by firstly removing symmetry, but doing so in a non-topological way. Hence,

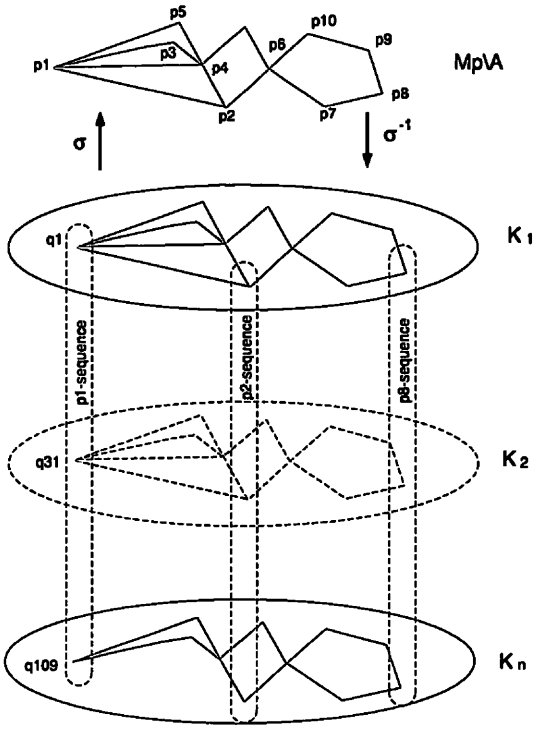


Figure 5.12: Symmetric closure example

we cannot refer to strong or observational equivalence as some sort of homomorphism or homeomorphism — they are very much behavioural concepts. This necessitates an alternative framework for analysing machine reduction in the structural context.

To do this we invoke the notion of *structural entropy*, or using Shannon's terminology [212], the information content of a structure [256]. Once again we look to graph theory where the concept of a graph entropy measure already exists Def 5.4.1 [177, 58].

5.4.1 Definition

The Graph Entropy Measure (GEM) of a graph G with vertex set $V(G)$ is defined as $\eta(G) = -\Sigma_{i=1}^{k} \rho_i \ln \rho_i$, where $\rho_i = \frac{|O_i|}{n}$ and $\mid V(G) \mid = \Sigma_{i=1}^{k} \mid O_i \mid = n$. O_i are the orbits generated by the group (in this case the automorphism group $\Gamma(G)$) action on

G. The orbits partition the vertex set into k subsets.

We will not dwell on the technicalities of the GEM here, suffice to say that it essentially provides a measure of *symmetry*. Hence, a complete graph will always have $\eta(G) = 0$ (minimum entropy and information), whilst a completely asymmetric graph will have $\eta(G) = \ln \frac{1}{n}$ (maximum entropy and information).

The relevance here is that machine reduction will tend to increase the entropy of the associated graph, and what is required is a technique which can unfold the structure in order to maximise symmetry (and hence minimise entropy). It is from this that the notion of *symmetric closure* emerged.

Symmetric closure will now be formally defined. The basic principle is to enforce the symmetry properties of the GCP, and to regenerate the lost states and actions on the basis that they need to be there in order that the expected symmetries manifest themselves. Hence gaps in the structure are 'closed' using symmetry as the guiding principle, thus the name. We will now be more formal.

5.4.2 Definition

Let **Sym** denote the symmetric closure transform, then a machine $\mathcal{M}_q$ to which symmetric closure has been applied is denoted by $\mathbf{Sym}(\mathcal{M}_q)$.

From Def 5.3.1 we know that if the respective graphs are relatively prime, then the symmetry group of the GCP will be the group product of the relevant symmetry groups. As mentioned above, this intrinsic symmetry is destroyed by communication and by machine reduction. Tau splitting addresses communication based asymmetry, symmetric closure reduction. The aim is to transform the given equation, being based on strong and observational equivalence, and to convert it to a form whereby the the quotient algorithm can be applied. But the two transforms are not independent, since we know from experiments that reduction can have a significant influence on the τ-

splitting transform Υ. That this is so is perhaps not surprising given that Υ itself is symmetry based. Hence, there is an ordering to the transforms — symmetric closure must be applied first. The steps are thus :-

$$\begin{aligned}
(\mathcal{M}_p \mid \mathcal{M}_r) \wr A\backslash B &\sim \mathcal{M}_q \text{ (call this } \mathcal{G}_{\mathcal{M}_q}) \xrightarrow{\mathbf{Sym}} \\
(\mathbf{Sym}(\mathcal{M}_p) \mid \mathcal{M}_r) \wr A\backslash B &\stackrel{S}{=} \mathbf{Sym}(\mathcal{M}_q) \text{ (call this } \mathcal{G}'_{\mathcal{M}_q}) \xrightarrow{\Upsilon} \\
(\mathbf{Sym}(\mathcal{M}_p) \mid \mathcal{M}_r)\backslash B &\stackrel{S}{=} \Upsilon(\mathbf{Sym}(\mathcal{M}_q)) \text{ (call this } \mathcal{G}''_{\mathcal{M}_q}) \\
\eta(\mathcal{G}_{\mathcal{M}_q}) \geq \eta(\mathcal{G}'_{\mathcal{M}_q}) &\geq \eta(\mathcal{G}''_{\mathcal{M}_q})
\end{aligned}$$

As we move from the first equation to the last, entropy is decreasing while symmetry is increasing. The first step is to remove the effects of reduction using symmetric closure **Sym** (Def 5.4.3). Once this lost structure has been restored the τ-splitting algorithm Υ can be applied. The effect of these two operations is to regenerate the symmetries lost via machine reduction and communication. Once completed, the process of quotient extraction can be carried out, from which the required solution will emerge.

5.4.3 Definition

Given two machines $\mathcal{M}_p$, $\mathcal{M}_q$ let $\mathcal{G}_{\mathcal{M}_p}$, $\mathcal{G}_{\mathcal{M}_q}$, $R(p)$, $R(q)$, $\Lambda(p)$, $\Lambda(q)$ and trace(p), trace(q) be the associated graphs, state reachability sets, alphabets and traces respectively. Further, let Γ_p and Γ_r be the respective symmetry groups. Let n_p and n_q be the number of states in $\mathcal{M}_p$ and $\mathcal{M}_q$. We define the sequences $\xi_{p,q} = q_1 \stackrel{S}{=} \ldots \stackrel{S}{=} q_n$ to be *complete* if n is equal to the number of $\tilde{\Pi}$-planes, otherwise it is said to be *incomplete.* An incomplete sequence is transformed into a complete sequence via algorithm 5.4. Note $q_i \stackrel{S}{=} q_j \Leftrightarrow R(q_i) = R(q_j)$, $\Lambda(q_i) = \Lambda(q_j)$, trace$(q_i)$ = trace(q_j). In terms of the $\tilde{\Pi}$-planes, if all the ξ-sequences are complete, then $\tilde{\Pi}_i \stackrel{S}{=} \tilde{\Pi}_j \stackrel{S}{=} \mathcal{M}_p - A \cup \{\tau_c\}\ \forall\ i,j$ else $\exists m,n : \tilde{\Pi}_m \stackrel{S}{\neq} \tilde{\Pi}_n$ and $\bigcup_i \xi_{q,p_i} = \bigcup_j \tilde{\Pi}_j$. Given a sequence of $\tilde{\Pi}$-planes $\tilde{\Pi}_1 \stackrel{S}{=} \tilde{\Pi}_2 \stackrel{S}{=} \ldots \stackrel{S}{=} \tilde{\Pi}_m \sim \tilde{\Pi}_n \stackrel{S}{=} \ldots \stackrel{S}{=} \tilde{\Pi}_r$, we must find a set Z of actions and states such

that $\tilde{\Pi}_1 \stackrel{S}{=} \tilde{\Pi}_2 \stackrel{S}{=} \ldots \stackrel{S}{=} \tilde{\Pi}_m \cup Z \stackrel{S}{=} \tilde{\Pi}_n \cup Z \stackrel{S}{=} \ldots \stackrel{S}{=} \tilde{\Pi}_r$. In terms of the composed machines, assume that under bisimulation $\mathcal{M}_p \mid \mathcal{M}_r$ has degenerate states, and thus can be reduced. Call this structure, its graph, and symmetry group $\overline{\mathcal{M}_p \mid \mathcal{M}_r}$, $\bar{G}$ and $\bar{\Gamma}$ respectively. From Theorem 5.3.2 we know that the underlying graph of this is $\mathcal{G}_{\mathcal{M}_p} \times \mathcal{G}_{\mathcal{M}_r}$, and that the symmetry group of the GCP, $\Gamma_{GCP} \subseteq \Gamma_p \otimes \Gamma_r$. In general, $\bar{\Gamma} \not\subseteq \Gamma_{GCP} \subseteq \Gamma_p \otimes \Gamma_r$. By Def 5.3.10 and Def 6.2.5, each Π-plane, ignoring all τ_c actions, is structurally equivalent to $\mathcal{M}_p$ with actions in A removed. Consider the states $q_i \in \mathcal{M}_q$, where i ranges over the number of distinct Π-planes, such that $\sigma(q_i) = p, p \in \mathcal{M}_p$ (each q_i is projected onto the same p). Then $q_1 \stackrel{S}{=} q_2 \stackrel{S}{=} \ldots \stackrel{S}{=} q_i$. If $\exists q' \in \Pi' : q' \stackrel{S}{\neq} q_1 \in \xi = q_1 \stackrel{S}{=} \ldots \stackrel{S}{=} q_i$, then transform Π' accordingly by adding a new state q'' or by the addition of a new action. This is repeated until $\tilde{\Pi}_1 \stackrel{S}{=} \tilde{\Pi}_2 \stackrel{S}{=} \ldots \stackrel{S}{=} \tilde{\Pi}_i \stackrel{S}{=} \mathcal{M}_p \backslash A$.

A simple algorithm for the above is :-

Repeat

For i = 1 to n_q do

construct $\{q_i\} = R(q_i), \Lambda(q_i), \text{trace}(q_i)$

For j = 1 to n_p do

construct $\{p_j\} = R(p_j), \Lambda(p_j), \text{trace}(p_j)$

if $\exists$ unique $p_k : \{q_i\} = \{p_k\}$ then

$q_i \in \xi_{q,p_k}$

else if $\mid \xi_{q,p_k} \mid \neq n_r$ then heuristics required

else

apply Z

calculate GEM

end do

end do

Until $\eta = 0$ or minimum

Essentially the algorithm states that if there are to many p's associated with a q, then there is to much symmetry and a (heuristic) choice needs to be made. If there are to many q's associated with a p, then reduction has occurred and new actions and states need to be added.

Thus symmetric closure is equivalent to constructing n distinct sequences of structurally equivalent q's, where $n = \mid \mathcal{M}_p \mid$. This is illustrated in figure 5.12. Here we have three sequences, only one of which is structurally equivalent namely the p_8-sequence. Both the p_1 and p_2-sequences need actions added in Π_n.

It should be clear from this definition that whilst symmetric closure will always produce a symmetric structure, it may not result in the correct solution. To see this, consider what happens if a whole Π-plane is lost via the reduction process, in which

case we may have $\eta(\bar{G}) = \eta(G)$. As a result we may have no indication that reduction has taken place, and to much information will have been lost to recover the required structure. The solution algorithm may then attempt to solve the equation when in fact no possible $\mathcal{M}_\tau$ will exist. However, this can be viewed as rather pathological in nature, and the assertion is that many problems will contain sufficient information to at least unfold strongly equivalent reduction, and even observational reduction, though the later is in general significantly more problematic. This is discussed in more detail in chapter 8 and section 5.6.

5.5 The algorithm

This section defines, in detail, the procedures and algorithms to solve the interface equation using the theory so far described. The assumption is that the given an interface equation is in the form,

$$
\begin{aligned}
&(\mathcal{M}_p \parallel \mathcal{M}_\tau) \wr A \backslash B \sim \mathcal{M}_q \quad \text{where} \\
&\Lambda(p) \cap \bar{\Lambda}(q) \subseteq \{\tau\}, \\
&\Lambda(p) \cap \overline{A} = \emptyset, \\
&\Lambda(q) \cap (A \cup \overline{A}) = \emptyset, \\
&\Lambda(r) \cap \Lambda(p) \subseteq \{\tau\}
\end{aligned}
$$

5.5.1 Step 1a

The first step in the process is to identify the states of $\mathcal{M}_p$ and $\mathcal{M}_q$ with $\mathcal{M}_\tau$. This is done using the procedure discussed earlier. We first derive the K-sets as per Martin's algorithm [140]. The σ mapping defined in Def 5.3.11 is then applied to derive the Π-planes, which are submachines of $\mathcal{M}_p$. This re-labeled $\mathcal{M}_q$ will be referred to as $\mathcal{M}'_q$.

5.5.2 Step 1b

It has been found that the application of symmetric closure is best done interactively with Step 1a, since the derivation of the K-sets provides vital clues as to where reduction has taken place. The best indication that symmetry conditions are being violated is the appearance of a distinct q state in more than one K-set, say $K(r_1)$ and $K(r_3)$. Symmetric closure 'unfolds' such states and via the addition of new states and actions, restores the symmetries. In practice symmetric closure will not always be successful, as discussed in section 5.4.

5.5.3 Step 2

Construction of the Π-planes is achieved by first grouping the states of $\mathcal{M}'_q$ with respect to the r component. Our aim here is to derive the structure of the embedded $\mathcal{M}_p$ machine in $\mathcal{M}_q$. The algorithms used by Shields and Martin to derive the K-sets can be used here with some modification see Def 5.3.10, such as the exclusion of the O-completeness condition. It is felt that graph partitioning has a key role to play here, but is left as a subject for future research. The output from this process is a set of K-sets which are all individually non-A/B reachable (the associated graphs are strongly connected).

5.5.4 Step 3

As discussed earlier, the connectedness conditions for the algorithm proposed here are weaker (more general) than those of preceding methods in that strong connectedness is not a precondition. The $\tilde{\Pi}(r)(r)$-sets, defined in Def 5.3.12, are constructed by forming the union of all $K(r)$-sets for each r. Ideally, the associated graph of each $\tilde{\Pi}$-plane will be weakly connected. If not the two methods referred to in this chapter can be used. We will assume here that the internal structure of each $\tilde{\Pi}$-plane is indeed

weakly connected.

Notice that at this stage we can begin to use symmetry as a validation tool. The emergent structure should contain certain symmetries, which if not there, are a good indicator to the existence of an error.

5.5.5 Step 4

The L sets (Def 5.3.13), are formed in exactly the same fashion as the Π-planes. However, unlike the Π-planes, it is assumed that each L-set will be non-A/B reachable.

5.5.6 Step 5

Let $\mu \in \Lambda(p) - (A \cup \overline{A} \cup B)$, an action of $\mathcal{M}_p$ which does not communicate. Let C be the set of all such actions. These actions provide no additional information in terms of the derivation of the $\mathcal{M}_r$ machine. Construct the machine $\mathcal{M}_q'' = \mathcal{M}_q' \backslash C$. $\mathcal{M}_q''$ will now contain non-τ actions which are from $\mathcal{M}_r$, and τ actions which are either intrinsic to $\mathcal{M}_p$ or $\mathcal{M}_r$, or are communicating τ actions. The primary function for removing the C actions is to reduce the processing requirements for quotient extraction. The next step is to distinguish between the τ_i and τ_c actions.

5.5.7 Step 6

The main steps in distinguishing between τ_i and τ_c actions is as follows:-

$$\begin{aligned}
&\textbf{Pred} = \exists \mu \in \Lambda(p) \cap A : p \rightarrow^{\mu} p' \text{ and} \\
&\mu \notin \Lambda(q) \text{ and } \neg\exists \tau : L(p'') \rightarrow^{\tau} L(p'''),\ \forall\, r \in R(r) \text{ for some} \\
&r \text{ or } \neg\exists \tau : \tilde{\Pi}(r) \rightarrow^{\tilde{\Pi}} (r'')\ \forall\, p \in R(p)
\end{aligned}$$

For each $(p,q,r) \rightarrow^{\tau} (p',q',r')$ transition **do**

If Pred = true **then**

Split τ (Step 7)

else

Mark intrinsic

end if

Repeat until $\tilde{\Pi} \stackrel{S}{=} \mathcal{M}_p\,(\Delta(\mathcal{M}_p, \tilde{\Pi}) = \emptyset)$

The rather complex expression within the **if** statement first checks to see if $\mathcal{M}_p$ did an action in A which $\mathcal{M}_q''$ does not do. This action will be involved in communication and hence will be associated with a τ_c action. However, even if such an action can be found there are situations where this may not be the correct τ action. Hence, the rest of the expression checks the symmetry properties to see if there are τ transitions in the $R(p) \times R(q)$ or $R(r) \times R(q)$ planes which are symmetric.

5.5.8 Step 7

From Step 6 identify all $(p,q,r) \rightarrow^{\tau_c} (p',q',r')$ transitions. Identify $\mu \in \Lambda(p) \cap A$: $p \rightarrow^{\mu} p'$ in $\mathcal{M}_p$. Add transition to $\mathcal{M}_q''$ $\forall\ r \in R(r)$. Create $r \rightarrow^{\overline{\mu}} r'$ transition and add to $\mathcal{M}_q''$ $\forall\ p \in R(p)$. Remove $(p,q,r) \rightarrow^{\tau_c} (p',q',r')$ from $\mathcal{M}_q''$ as described in Def 5.3.14. An $\mathcal{M}_q$ with all its τ_c actions split is labeled $\mathcal{M}_q^{\bullet}$.

5.5.9 Step 8

We now apply the quotient algorithm.

1. Factor $\mathcal{G}_{\mathcal{M}_q}$ (a weakly connected digraph)
2. $\mathcal{G}_{\mathcal{M}_q} := \mathbf{U}(D(V_q, E_q))$ the underlying graph of $D(V_q, E_q)$.

3. $\mathcal{G}_{\mathcal{M}_p} := \mathrm{U}(D(V_p, E_p))$ the underlying graph of $D(V_p, E_p)$.

4. Find the prime factorisation of $\mathcal{G}_{\mathcal{M}_q}$ (say m-components).

5. Let $A_1, A_2, \ldots, A_m$ be the arc classes corresponding to the edge classes $E_1, E_2, \ldots, E_m$ (associated with $\mathcal{G}_{\mathcal{M}_q}$).

6. Set $R = \emptyset$.

7. Apply algorithm [69].

   for each pair of arc classes A_i, A_j

   for each pair of adjacent copies G_i^x, G_i^y

   for each pair of edges $u - v \in E_i, u - u' \in E_j, u \in G_i^x, u' \in G_i^y$

   if the 4-cycle $u - u' - v' - v - u$ lifts to a subgraph of $D(V_q, E_q)$ that shows a conflict between A_i and A_j

   then $R := R \cup \{(A_i, A_j)\}$;

   R^* := the reflexive, transitive closure of R;

   p := the number of equivalence classes R^*;

   for i := 1 to p

   begin

   Let $\{A_i\}$ be the ith class in R^*;

   Let $\{E_i\}$ be the corresponding edge classes in $\mathcal{G}_{\mathcal{M}_q}$;

   $H := G_1^x \cup \cdots \cup G_i^x$ for some $x \in V(G)$;

   Let D_i be the subgraph of $D(V_q, E_q)$ to which H lifts;

   The ith prime factor of $D(V_q, E_q)$ is isomorphic to D_i;

   end

8. $D(V_q, E_q) = D_1 \times D_2 \times \cdots \times D_m$

9. Repeat for $\mathcal{G}_{\mathcal{M}_p} = D(V_p, E_p)$, giving $D(V_p, E_p) = D_1 \times D_2 \times \cdots \times D_n$, where $m \geq n$.

The assertion is that $\mathcal{M}_r = D(V_r, E_r)$ is equal to :-

$$\begin{aligned} D(V_r, E_r) &\cong \frac{D(V_q, E_q)}{D(V_p, E_p)} \\ &\cong \frac{D_1 \times D_2 \times \cdots \times D_m}{D_1 \times D_2 \times \cdots \times D_n} \\ &\cong D_{m-n} \times D_{m-n+1} \times \cdots \times D_m \end{aligned}$$

Hence the unlabeled $\mathcal{G}_{\mathcal{M}_r} \cong D_{m-n} \times D_{m-n+1} \times \cdots \times D_m$. This is the unlabeled underlying digraph of $\mathcal{M}_r$.

5.5.10 Step 9

The output from Step 8 is the underlying digraph of $\mathcal{M}_r$ which has unlabeled edges, but resides within $\mathcal{M}_q''$. Labeling of the edges, which corresponds to actions in the machine representation, can be achieved by pattern matching on $\mathcal{M}_q''$, or via heuristic methods.

5.5.11 Step 10

Use tool such as Concurrency Workbench to check that $\mathcal{M}_r$ is a solution of $(\mathcal{M}_p \backslash C \mid \mathcal{M}_r) \wr A \backslash B \stackrel{S}{=} \mathcal{M}_q'' \Rightarrow (\mathcal{M}_p \parallel \mathcal{M}_r) \wr A \backslash B \stackrel{S}{=} \mathcal{M}_q$.

It is interesting to note that under certain conditions the quotient graph algorithm is not needed, and we can extract the quotient via indirect means using the L-sets. Clearly enormous processing gains could be made here, but this is a matter for further research (see chapter 8).

5.6 Validation

The preceding theoretical discussions have laid the foundation for the graph theoretic approach. In this chapter a number of pathological examples are examined which were designed to test the theories limits. A criticism of other examples in this thesis could be that the author had prior knowledge of the answer. This has been necessary to avoid what could become a tedious process of trial and error before a problem where a solution was even possible could be found. In this chapter the examples were generated by Dr G. Martin, who understanding the theory, could devise examples which could probe its limits of application, and for which the author would have no prior knowledge of the solution. This was felt to be an important condition for formal validation. All the examples involved have an element of reduction via strong equivalence, since although the theory is based on the notion of structural equivalence many, indeed most, of strong equivalence have been found to be solvable. This issue is formalised to some extent in section 5.4, where an interesting relationship is drawn between reduction, symmetry and entropy.

The chapter centres on three examples. These examples, with the exception of example # 1, are discussed not in terms of the details of actually applying the algorithm, but rather the issues that arose from the solution process.

5.6.1 Example 1

The algorithm is now applied to a small example (see appendix B for full specification). The aim of this section is to apply the theory described to the interface equation and to generate the $\mathcal{M}_r$ machine from the given $\mathcal{M}_p$ and $\mathcal{M}_q$. All solutions were checked using CWB [156].

Step 1a and 1b

Form K-sets, Π-planes and $\tilde{\Pi}$-planes, and apply symmetric closure. Since the equation is already in a structurally equivalent form, symmetric closure is not required.

Form triples :-

$$
\begin{aligned}
\sigma &: R(q) \longrightarrow R(p) \times R(q) \times R(r) \\
\sigma &: q_1 \mapsto (p_1, q_1, r_1)\ p_1 \mid r_1 \stackrel{S}{=} q_1 \\
&: q_2 \mapsto (p_2, q_2, r_1)\ p_2 \mid r_1 \stackrel{S}{=} q_2 \\
&: q_3 \mapsto (p_3, q_3, r_1)\ p_3 \mid r_1 \stackrel{S}{=} q_3 \\
&: q_4 \mapsto (p_4, q_4, r_1)\ p_4 \mid r_1 \stackrel{S}{=} q_4 \\
&: q_5 \mapsto (p_5, q_5, r_1)\ p_5 \mid r_1 \stackrel{S}{=} q_5 \\
&: q_6 \mapsto (p_1, q_6, r_2)\ p_1 \mid r_2 \stackrel{S}{=} q_6 \\
&: q_7 \mapsto (p_1, q_7, r_3)\ p_1 \mid r_3 \stackrel{S}{=} q_7 \\
&: q_8 \mapsto (p_1, q_8, r_4)\ p_1 \mid r_4 \stackrel{S}{=} q_8 \\
&: q_9 \mapsto (p_2, q_9, r_2)\ p_2 \mid r_2 \stackrel{S}{=} q_9 \\
&: q_{10} \mapsto (p_2, q_{10}, r_3)\ p_2 \mid r_3 \stackrel{S}{=} q_{10} \\
&: q_{11} \mapsto (p_2, q_{11}, r_4)\ p_2 \mid r_4 \stackrel{S}{=} q_{11} \\
&: q_{12} \mapsto (p_3, q_{12}, r_2)\ p_3 \mid r_2 \stackrel{S}{=} q_{12} \\
&: q_{13} \mapsto (p_3, q_{13}, r_3)\ p_3 \mid r_3 \stackrel{S}{=} q_{13} \\
&: q_{14} \mapsto (p_3, q_{14}, r_4)\ p_3 \mid r_4 \stackrel{S}{=} q_{14} \\
&: q_{15} \mapsto (p_4, q_{15}, r_2)\ p_4 \mid r_2 \stackrel{S}{=} q_{15} \\
&: q_{16} \mapsto (p_4, q_{16}, r_3)\ p_4 \mid r_3 \stackrel{S}{=} q_{16} \\
&: q_{17} \mapsto (p_4, q_{17}, r_4)\ p_4 \mid r_4 \stackrel{S}{=} q_{17} \\
&: q_{18} \mapsto (p_5, q_{18}, r_2)\ p_5 \mid r_2 \stackrel{S}{=} q_{18}
\end{aligned}
$$

$$: \quad q_{19} \mapsto (p_5, q_{19}, r_3)\ p_5 \mid r_3 \stackrel{S}{=} q_{19}$$

$$: \quad q_{20} \mapsto (p_5, q_{20}, r_4)\ p_5 \mid r_4 \stackrel{S}{=} q_{20}$$

$$: \quad q_{21} \mapsto (p_4, q_{21}, r_5)\ p_4 \mid r_5 \stackrel{S}{=} q_{21}$$

$$: \quad q_{22} \mapsto (p_3, q_{22}, r_5)\ p_3 \mid r_5 \stackrel{S}{=} q_{22}$$

$$: \quad q_{23} \mapsto (p_1, q_{23}, r_5)\ p_1 \mid r_5 \stackrel{S}{=} q_{23}$$

$$: \quad q_{24} \mapsto (p_2, q_{24}, r_5)\ p_2 \mid r_5 \stackrel{S}{=} q_{24}$$

$$: \quad q_{25} \mapsto (p_5, q_{25}, r_5)\ p_5 \mid r_5 \stackrel{S}{=} q_{25}$$

Step 2

$\mathcal{M}_p$ is non-A/B reachable (in this example $B = \emptyset$). This is determined via the reachability matrix :-

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 3 & 0 & 1 & 4 & 2 \\ 2 & 1 & 0 & 3 & 1 \\ 4 & 1 & 2 & 0 & 3 \\ 1 & 2 & 3 & 2 & 0 \end{pmatrix}$$

Hence the Π-planes are :-

$$\Pi(r_1) = \{(p_1, q_1, r_1), (p_2, q_2, r_1), (p_3, q_3, r_1), (p_4, q_4, r_1), (p_5, q_5, r_1)\}$$

$$\Pi(r_2) = \{(p_1, q_6, r_2), (p_2, q_9, r_2), (p_3, q_{12}, r_2), (p_4, q_{15}, r_2), (p_5, q_{18}, r_2)\}$$

$$\Pi(r_3) = \{(p_1, q_7, r_3), (p_2, q_{10}, r_3), (p_3, q_{13}, r_3), (p_4, q_{16}, r_3), (p_5, q_{19}, r_3)\}$$

$$\Pi(r_4) = \{(p_1, q_8, r_4), (p_2, q_{11}, r_4), (p_3, q_{14}, r_4), (p_4, q_{17}, r_4), (p_5, q_{20}, r_4)\}$$

$$\Pi(r_5) = \{(p_4, q_{21}, r_5), (p_3, q_{22}, r_5), (p_1, q_{23}, r_5), (p_2, q_{24}, r_5), (p_5, q_{25}, r_5)\}$$

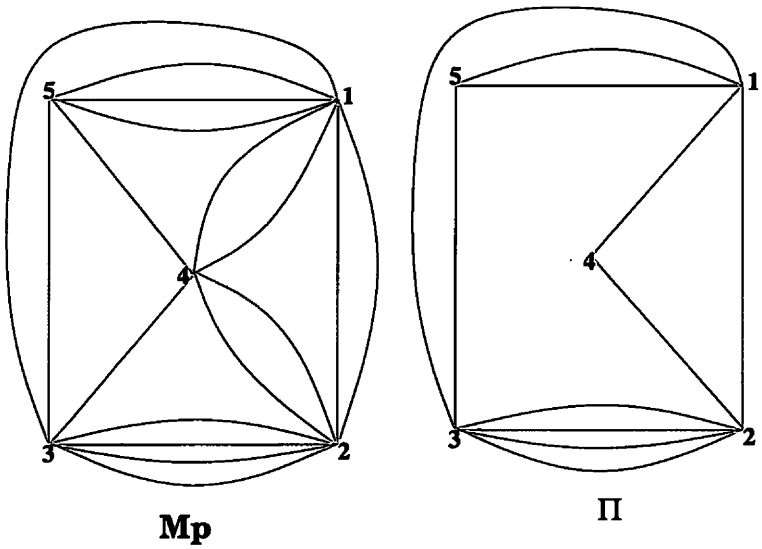


Figure 5.13: $\mathcal{G}_{\mathcal{M}_p}$ and $\tilde{\Pi}$

Step 3

It is clear from Step 2, that in this case, $\Pi(r_i) = \tilde{\Pi}(r_i)\ \forall\ r_i$ A comparison of the $\tilde{\Pi}$ and $\mathcal{G}_{\mathcal{M}_p}$ structures is shown in figure 5.13.

Step 4

The L-sets are :-

$$
\begin{aligned}
L(p_1) &= \{(p_1, q_1, r_1), (p_1, q_6, r_2), (p_1, q_7, r_3), (p_1, q_8, r_4), (p_1, q_{23}, r_5)\} \\
L(p_2) &= \{(p_2, q_2, r_1), (p_2, q_9, r_2), (p_2, q_{10}, r_3), (p_2, q_{11}, r_4), (p_2, q_{24}, r_5)\} \\
L(p_3) &= \{(p_3, q_3, r_1), (p_3, q_{12}, r_2), (p_3, q_{13}, r_3), (p_3, q_{14}, r_4), (p_3, q_{22}, r_5)\} \\
L(p_4) &= \{(p_4, q_4, r_1), (p_4, q_{15}, r_2), (p_4, q_{16}, r_3), (p_4, q_{17}, r_4), (p_4, q_{21}, r_5)\} \\
L(p_5) &= \{(p_5, q_5, r_1), (p_5, q_{18}, r_2), (p_5, q_{19}, r_3), (p_5, q_{20}, r_4), (p_5, q_{25}, r_5)\}
\end{aligned}
$$

Step 5

The set $C = \{\alpha, \beta, \gamma\}$. These actions are removed from $\mathcal{M}_q$ to give the structure shown in figure 5.14, i.e. $\mathcal{M}_q^{\bullet} \backslash C$.

Step 6

The algorithm classifies the τ actions as :-

Intrinsic — The following transitions are intrinsic τ actions :-

$$T_i = \{q_{21} \rightarrow^{\tau_i} q_{16}, q_{23} \rightarrow^{\tau_i} q_7, q_{25} \rightarrow^{\tau_i} q_{19}, q_{22} \rightarrow^{\tau_i} q_{13},$$
$$q_{24} \rightarrow^{\tau_i} q_{10}, q_1 \rightarrow^{\tau_i} q_5, q_6 \rightarrow^{\tau_i} q_{18}, q_7 \rightarrow^{\tau_i} q_{19},$$
$$q_8 \rightarrow^{\tau_i} q_{20}, q_{23} \rightarrow^{\tau_i} q_{25}, q_3 \rightarrow^{\tau_i} q_2, q_{12} \rightarrow^{\tau_i} q_9,$$
$$q_{13} \rightarrow^{\tau_i} q_{10}, q_{14} \rightarrow^{\tau_i} q_{11}, q_{22} \rightarrow^{\tau_i} q_{24}\}$$

Communicating — The following transitions are communicating τ actions :-

$$T_c = \{q_9 \rightarrow^{\tau_c} q_7, q_9 \rightarrow^{\tau_c} q_8, q_{15} \rightarrow^{\tau_c} q_{19}, q_{15} \rightarrow^{\tau_c} q_{20},$$
$$q_{16} \rightarrow^{\tau_c} q_{22}, q_{16} \rightarrow^{\tau_c} q_{23}, q_{18} \rightarrow^{\tau_c} q_7, q_{18} \rightarrow^{\tau_c} q_8,$$
$$q_{21} \rightarrow^{\tau_c} q_3, q_{21} \rightarrow^{\tau_c} q_1, q_{24} \rightarrow^{\tau_c} q_{13}, q_{24} \rightarrow^{\tau_c} q_4\}$$

Remember, intrinsic tau actions will bijectively map $\tilde{\Pi}$-sets to $\tilde{\Pi}$-sets or L-sets to L-sets. While communicating tau actions can do the same, they will be identified by missing actions from $\mathcal{M}_p$. The graph of $\mathcal{M}_q \backslash C \backslash \{\tau_i\}$ is shown in figure 5.15.

Step 7

The τ_c actions are now split.

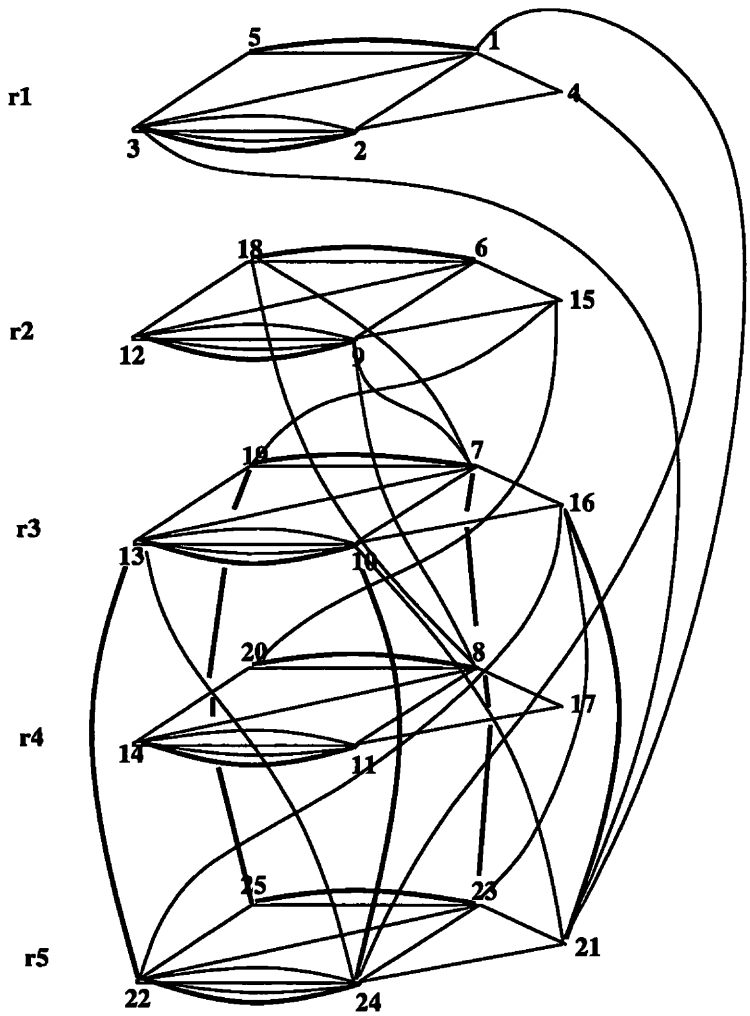


Figure 5.14: $p\tau$-graph

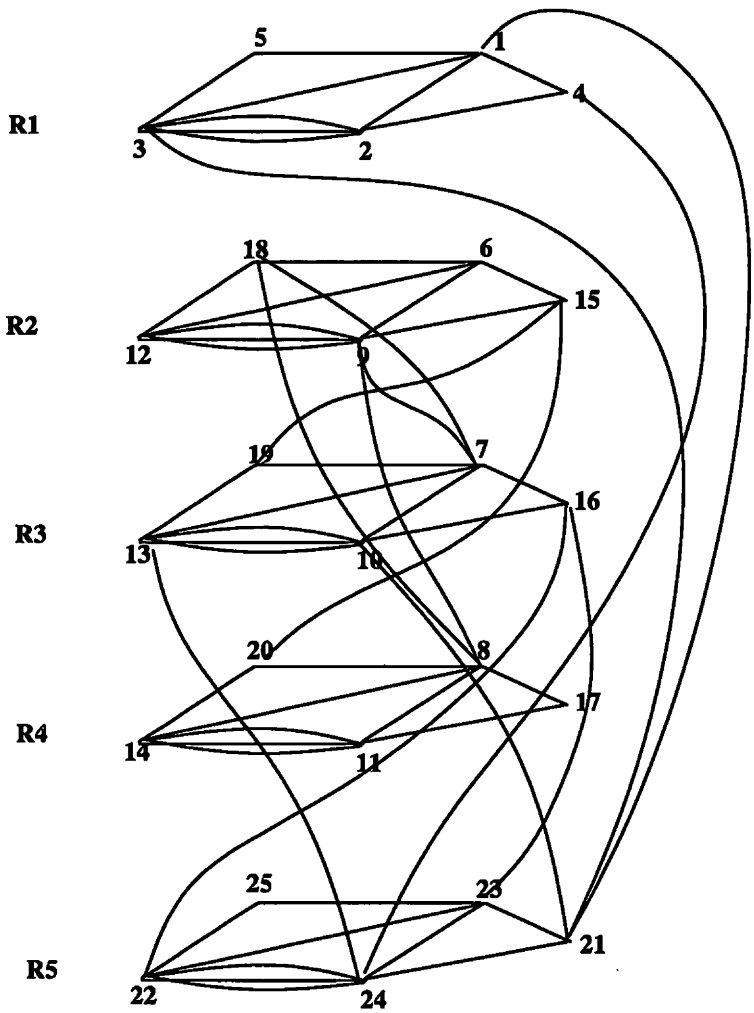


Figure 5.15: $p\tau$-graph with τ_i actions removed

1 From figure 5.13 $\mathcal{M}_p$ p_5 does a $\overline{send2}$ to p_1. This action is missing from $\tilde{\Pi}$ in figure 5.14. However, we note that, via projection, that the machine $\mathcal{M}_q$ does a τ transition from q_{18} to q_7 ($\Pi(r_2)$-plane to the $\Pi(r_3)$-plane). It also does a q_{18} to q_8 ($\Pi(r_2)$-plane to the $\Pi(r_4)$-plane).

2 From figure 5.13 $\mathcal{M}_p$ p_4 does a $\overline{send1}$ to p_1. This action is missing from $\tilde{\Pi}$ in figure 5.14. However, we note that, via projection, that the machine $\mathcal{M}_q$ does a τ transition from q_{21} to q_1 ($\Pi(r_5)$-plane to the $\Pi(r_1)$-plane). It also does a q_{16} to q_{23} ($\Pi(r_3)$-plane to the $\Pi(r_5)$-plane).

3 From figure 5.13 $\mathcal{M}_p$ p_2 does a $\overline{send1}$ to p_4. This action is missing from $\tilde{\Pi}$ in figure 5.14. However, we note that, via projection, that the machine $\mathcal{M}_q$ does a τ transition from q_{10} to q_{21} ($\Pi(r_3)$-plane to the $\Pi(r_5)$-plane). It also does a q_{24} to q_4 ($\Pi(r_5)$-plane to the $\Pi(r_1)$-plane).

4 From figure 5.13 $\mathcal{M}_p$ p_2 does a $\overline{send2}$ to p_1. This action is missing from $\tilde{\Pi}$ in figure 5.14. However, we note that, via projection, that the machine $\mathcal{M}_q$ does a τ transition from q_9 to q_7 ($\Pi(r_2)$-plane to the $\Pi(r_3)$-plane). It is also related to a q_9 to q_8 ($\Pi(r_2)$-plane to the $\Pi(r_4)$-plane).

5 From figure 5.13 $\mathcal{M}_p$ p_4 does a $\overline{send1}$ to p_3. This action is missing from $\tilde{\Pi}$ in figure 5.14. However, we note that, via projection, that the machine $\mathcal{M}_q$ does a τ transition from q_{16} to q_{22} ($\Pi(r_3)$-plane to the $\Pi(r_5)$-plane).

6 From figure 5.13 $\mathcal{M}_p$ p_4 does a $\overline{send2}$ to p_5. This action is missing from $\tilde{\Pi}$ in figure 5.14. However, we note that, via projection, that the machine $\mathcal{M}_q$ does a τ transition from q_{15} to q_{19} ($\Pi(r_2)$-plane to the $\Pi(r_3)$-plane). It is also related to a q_{15} to q_{20} ($\Pi(r_2)$-plane to the $\Pi(r_4)$-plane).

The process now is to split the τ_c actions via the τ-splitting transform. The τ-actions that need to be split are indicated in figure 5.15.

$$\Upsilon \quad (q_9 \rightarrow^{\tau_c} q_7)$$

$$\Upsilon \quad (q_9 \rightarrow^{\tau_c} q_8)$$

$$\Upsilon \quad (q_{10} \rightarrow^{\tau_c} q_{21})$$

$$\Upsilon \quad (q_{15} \rightarrow^{\tau_c} q_{19})$$

$$\Upsilon \quad (q_{15} \rightarrow^{\tau_c} q_{20})$$

$$\Upsilon \quad (q_{16} \rightarrow^{\tau_c} q_{22})$$

$$\Upsilon \quad (q_{16} \rightarrow^{\tau_c} q_{23})$$

$$\Upsilon \quad (q_{18} \rightarrow^{\tau_c} q_7)$$

$$\Upsilon \quad (q_{18} \rightarrow^{\tau_c} q_8)$$

$$\Upsilon \quad (q_{21} \rightarrow^{\tau_c} q_1)$$

$$\Upsilon \quad (q_{24} \rightarrow^{\tau_c} q_4)$$

The original actions, which will be an action from $\mathcal{M}_p$ and its dual from $\mathcal{M}_r$, are assigned to the appropriate vertices. Thus we gradually 'rebuild' $\tilde{\Pi}$ to the point where it is equivalent to $\mathcal{M}_p$. The process is terminated once $\tilde{\Pi}$ has been fully reconstructed, and is equivalent to the original $\mathcal{M}_p$ machine.

- From step [1] we deduce that r_2 does a $\overline{send2}$ to r_4, and that r_2 does a $\overline{send2}$ to r_3.
- From step [2] we deduce that r_3 does a $\overline{send1}$ to r_5, and r_5 does a $\overline{send1}$ to r_1.
- From step [3] we deduce that r_5 does a $\overline{send1}$ to r_1, and r_3 does a $\overline{send1}$ to r_5.
- From step [4] we deduce that r_2 does a $\overline{send2}$ to r_4, and r_2 does a $\overline{send2}$ to r_3.
- From step [5] we deduce that r_3 does a $\overline{send1}$ to r_5.

- From step [6] we deduce that r_2 does a $\overline{send2}$ to r_4, and r_2 does a $\overline{send2}$ to r_3.

The output from the τ-splitting transform Υ is as follows :-

$$\begin{aligned}
\Upsilon(q_9) &\Leftarrow < p_2 \rightarrow^{\overline{send2}} p_1 \parallel r_2 \rightarrow^{send2} r_3 > + < p_2 \rightarrow^{\overline{send2}} p_1 \parallel r_2 \rightarrow^{send2} r_4 > \\
\Upsilon(q_{10}) &\Leftarrow < p_2 \rightarrow^{\overline{send1}} p_4 \parallel r_3 \rightarrow^{send1} r_5 > \\
\Upsilon(q_{15}) &\Leftarrow < p_4 \rightarrow^{\overline{send2}} p_5 \parallel r_2 \rightarrow^{send2} r_3 > + < p_4 \rightarrow^{\overline{send2}} p_5 \parallel r_2 \rightarrow^{send2} r_4 > \\
\Upsilon(q_{16}) &\Leftarrow < p_4 \rightarrow^{\overline{send1}} p_1 \parallel r_3 \rightarrow^{send1} r_5 > + < p_4 \rightarrow^{\overline{send1}} p_3 \parallel r_3 \rightarrow^{send1} r_5 > \\
\Upsilon(q_{18}) &\Leftarrow < p_5 \rightarrow^{\overline{send2}} p_1 \parallel r_2 \rightarrow^{send2} r_3 > + < p_5 \rightarrow^{\overline{send2}} p_1 \parallel r_2 \rightarrow^{send2} r_4 > \\
\Upsilon(q_{21}) &\Leftarrow < p_4 \rightarrow^{\overline{send1}} p_1 \parallel r_5 \rightarrow^{send1} r_1 > \\
\Upsilon(q_{24}) &\Leftarrow < p_2 \rightarrow^{\overline{send1}} p_4 \parallel r_5 \rightarrow^{send1} r_1 >
\end{aligned}$$

The corresponding 'split' edges are added to $\mathcal{G}_{\mathcal{M}_q}$, along with a check that $\tilde{\Pi}$ is identical to $\mathcal{M}_p$. The structure of $\mathcal{M}_q^{\bullet}$ is given in figure 5.16.

Step 8

Apply quotient algorithm. We do not give details here to avoid excessive complexity. Suffice to say that manual application is complex and time consuming.

Step 9

The underlying digraph of $\mathcal{M}_r$ is shown in figure 5.17

Assignment of edge labels results in the following CCS prefix notation specification for $\mathcal{M}_r$:-

$$\begin{aligned}
r_1 &\Leftarrow \mu r_2 + \nu r_3 + \nu r_4 \\
r_2 &\Leftarrow \text{send2}r_3 + \text{send2}r_4 \\
r_3 &\Leftarrow \mu r_2 + \nu r_1 \\
r_4 &\Leftarrow \nu r_1 + \text{send1}r_5 \\
r_5 &\Leftarrow \tau r_4 + \nu r_4 + \text{send1}r_1
\end{aligned}$$

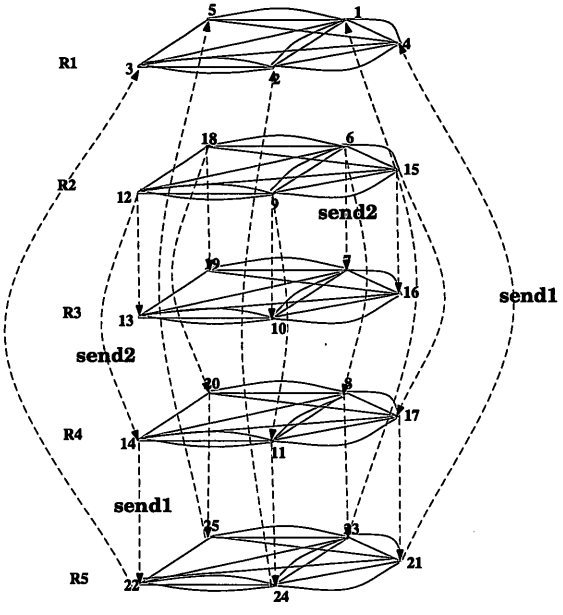


Figure 5.16: $p\tau$-graph with split τ_c actions

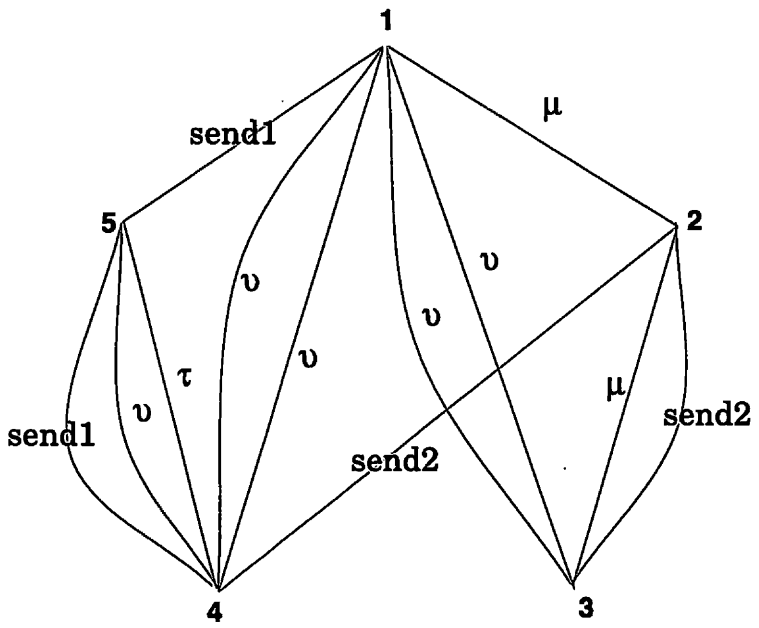


Figure 5.17: $\mathcal{G}_{\mathcal{M}_r}$

This structure, being the underlying graph for $\mathcal{M}_r$, will reside in G_q, and a pattern searching algorithm can now be applied in order to determine the labeling scheme. Alternatively the labeling could be carried out by inspection. In the later case, observe Λ_q and Λ_p, the alphabets of $\mathcal{M}_q$ and $\mathcal{M}_p$ respectively. Notice that $\mathcal{M}_p$ performs no ν and μ actions whilst $\mathcal{M}_q$ does. These actions must come from $\mathcal{M}_r$ and hence must be included. But how are these actions, and the associated event transitions, arrived at? The process is the same as that used from the intrinsic $\mathcal{M}_r$ τ-actions described earlier. What we should see in the $\mathcal{M}_q$ machine are ν and μ between r-planes. It is a relatively simple process to derive these. Indeed in the example above it can be done via inspection, though for real applications it is envisaged that the $\mathcal{M}_r$ structure will be derived first, onto which the relevant actions will be assigned. Once this procedure is carried out we arrive at the required solution. Notice that r_3 and r_4 have to be interchanged to arrive at the known solution. The labeled digraph

$\mathcal{G}_{\mathcal{M}_r} \cong D_{m-n} \times D_{m-n+1} \times \cdots \times D_m$ is converted to CCS notation via Theorem 5.3.2 to give $\mathcal{M}_r = \mathcal{M}_r^1 \mid \mathcal{M}_r^2 \mid \cdots \mid \mathcal{M}_r^{m-n}$.

Step 10

It is a relatively simple task on Concurrency Workbench to verify that $\mathcal{M}_r$ is a solution to the given interface equation.

5.6.2 Example 2

This example attempted to create confusion by having multiple τ_c actions and reducing $\mathcal{M}_q$ using strong equivalence.

$$
\begin{aligned}
p_1 &\Leftarrow \alpha p_2 + \tau p_6 + \delta p_6 + \epsilon p_6 + \eta p_6 \\
p_2 &\Leftarrow \beta p_3 \\
p_3 &\Leftarrow \gamma p_4 + \delta p_4 + \epsilon p_4 \\
p_4 &\Leftarrow \alpha p_5 \\
p_5 &\Leftarrow \beta p_1 \\
p_6 &\Leftarrow \alpha p_7 \\
p_7 &\Leftarrow \beta p_8 \\
p_8 &\Leftarrow \gamma p_9 + \delta p_9 + \epsilon p_9 \\
p_9 &\Leftarrow \delta p_{10} \\
p_{10} &\Leftarrow \beta p_6
\end{aligned}
$$

$$
\begin{aligned}
q_1 &\Leftarrow \alpha q_{22} + \tau q_3 + \tau q_{17} + \tau q_{25} + \tau q_2 + \tau q_{21} \\
q_2 &\Leftarrow \alpha q_{32} + \tau q_{17} \\
q_3 &\Leftarrow \alpha q_{24} + \tau q_2 \\
q_4 &\Leftarrow \beta q_1 + \tau q_{11}
\end{aligned}
$$

$$q_5 \Leftarrow \gamma q_9 + \tau q_{23} + \tau q_{34} + \tau q_{20}$$

$$q_6 \Leftarrow \beta q_{16} + \tau q_{32}$$

$$q_7 \Leftarrow \beta q_{17}$$

$$q_8 \Leftarrow \tau q_7 + \tau q_{12}$$

$$q_9 \Leftarrow \alpha q_4 + \tau q_{37}$$

$$q_{10} \Leftarrow \alpha q_{33} + \tau q_{25} + \tau q_2 + \tau q_{21}$$

$$q_{11} \Leftarrow \beta q_{21} + \tau q_{30}$$

$$q_{12} \Leftarrow \tau q_{14}$$

$$q_{13} \Leftarrow \gamma q_{34} + \tau q_{37} + \tau q_{20}$$

$$q_{14} \Leftarrow 0$$

$$q_{15} \Leftarrow \beta q_{18}$$

$$q_{16} \Leftarrow \gamma q_{27} + \tau q_{12} + \tau q_{28}$$

$$q_{17} \Leftarrow \alpha q_{19}$$

$$q_{18} \Leftarrow \gamma q_{23} + \tau q_{34}$$

$$q_{19} \Leftarrow \beta q_{26}$$

$$q_{20} \Leftarrow \gamma q_{37} + \tau q_{18} + \tau q_{23}$$

$$q_{21} \Leftarrow \alpha q_{35} + \tau q_2 + \tau q_{17} + \tau q_{36}$$

$$q_{22} \Leftarrow \beta q_5 + \tau q_{35}$$

$$q_{23} \Leftarrow \alpha q_{30}$$

$$q_{24} \Leftarrow \beta q_{31} + \tau q_{32}$$

$$q_{25} \Leftarrow \alpha q_6 + \tau q_2$$

$$q_{26} \Leftarrow \gamma q_{14} + \tau q_{27}$$

$$q_{27} \Leftarrow \tau q_{29} + \tau q_{12}$$

$$q_{28} \Leftarrow \gamma q_{12} + \tau q_{14} + \tau q_{26}$$

$$q_{29} \Leftarrow \beta q_2 + \tau q_7$$

$$q_{30} \Leftarrow \beta q_{36}$$

$$
\begin{aligned}
q_{31} &\Leftarrow \gamma q_8 + \tau q_{14} + \tau q_{27} + \tau q_{28} \\
q_{32} &\Leftarrow \beta q_{28} + \tau q_{19} \\
q_{33} &\Leftarrow \beta q_{13} + \tau q_{35} \\
q_{34} &\Leftarrow \alpha q_{38} + \tau q_{37} \\
q_{35} &\Leftarrow \beta q_{20} + \tau q_{15} \\
q_{36} &\Leftarrow \alpha q_{15} + \tau q_{25} + \tau q_2 + \tau q_{17} \\
q_{37} &\Leftarrow \alpha q_{11} + \tau q_{23} \\
q_{38} &\Leftarrow \beta q_{10} + \tau q_{11}
\end{aligned}
$$

The equation to be solved was :-

$$(\mathcal{M}_p \mid \mathcal{M}_r) \wr \{\eta, \epsilon, \delta, \} \sim \mathcal{M}_q$$

The following points emerged from the solution process, which took in total around 30 minutes of manual work.

1. The stage of the algorithm for which this example posed the greatest problem was the mapping of $\mathcal{M}_q$ to $\mathcal{M}_p$. While a mapping was achieved it was clear that had certain other conditions arose, e.g. fully symmetric traces where there is a large degree of redundancy, the labeling algorithm may have been unable to produce the required mapping. The question is was whether a fully symmetric problem could actually be constructed. All attempts had sufficient asymmetry, and hence information, to enable the algorithm to orient itself and provide a suitable mapping. This problem is linked to the reduction problem, in that there appears to be an interesting issue concerning symmetry. Symmetry is

central to the whole algorithm, yet it appears that a problem which contained to much symmetry would not be solvable!

2. The constraints imposed by the underlying symmetries provides a useful means of validating the given specifications. Indeed there were three key errors in the original specifications given to the author, all of which were identified by applying the symmetry rules. This was very encouraging.

3. Identification of the τ_i actions proved to be straightforward. Once again it was the underlying symmetry relationships that provide the primary lever. The value of this is that it in turn makes the τ_c actions easily identifiable.

4. The multiple τ_c actions proved to present no difficulties for the *tau*-splitting algorithm. What did transpire was that in general a number of solutions are possible and that the underlying state transition graph of these solutions are isomorphic. This was an unexpected result.

5. Reduction via strong equivalence was in this case not a problem, since the reduced states were small in number (only two). Examination of $\mathcal{M}_p$ and $\mathcal{M}_q$ suggested that the number of states in $\mathcal{M}_r$ was greater than or equal to 4. The assumption was that the solution contained 4 states, but it was conceivable that a whole Π-plane was missing and that actually it was a 5-state machine. The only technique so far developed to reverse the effects of reduction is symmetric closure. The success of this technique is critically dependent on the information lost through reduction (or increase in entropy). There appears to be a limit past which symmetric closure cannot derive the required structure. Of interest here is whether, assuming that it comes up with a structure that is symmetric and consistent with the given machines, a solution will be found.

5.6.3 Example 3

This example, again developed by Dr G. Martin, took the redundancy issue of the previous example further and increased the level of machine reduction. The small size was deliberate since it compounded the problem of information loss.

$$A = \{\delta, \epsilon, \eta\}$$

$$\begin{aligned} p_1 &\Leftarrow \alpha p_1 + \eta p_1 + \epsilon p_2 \\ p_2 &\Leftarrow \beta p_2 + \delta p_2 + \eta p_3 \\ p_3 &\Leftarrow \gamma p_3 + \epsilon p_3 + \delta p_2 \end{aligned}$$

$$\begin{aligned} q_1 &\Leftarrow \alpha q_1 + \tau q_2 \\ q_2 &\Leftarrow \beta q_2 + \tau q_3 \\ q_3 &\Leftarrow \gamma q_3 + \tau q_4 + \tau q_5 \\ q_4 &\Leftarrow \gamma q_4 + \tau q_2 \\ q_5 &\Leftarrow \beta q_5 + \tau q_6 \\ q_6 &\Leftarrow \gamma q_6 + \tau q_4 \end{aligned}$$

The algorithm was able to solve this example, but with some difficulty and required some heuristics to actually complete the process. The observations made (over and above those of the previous example) are as follows :-

1. The initial configuration of the problem suggested a 2-state $\mathcal{M}_\tau$-machine. In fact it should have been a 3-state machine. While it became clear that this was the case, the degree of information lost meant that an element of heuristics had to be applied during the symmetric closure process. The observation here is that it appears that if a whole Π-plane is effectively removed via reduction, the current symmetric closure process is to simplistic to recover the structure. Had $q_4 \Leftarrow \alpha q_4 + \tau q_2$ the evidence would have been overwhelming for a $2-$ state $\mathcal{M}_\tau$.

2. The example demonstrated once again the power of the symmetry approach, both in terms of validation and finding the intrinsic τ-actions for instance.

With all these examples (and the large example presented in appendix C), the algorithm has been followed as given. The author is confident therefore that the algorithm is a viable one and that overall, the graph theoretic approach does indeed solve the interface equation to a high degree of generality.

5.7 Summary and discussion

This chapter has presented a method to solve the interface equation using graph theory. Central to this approach is the use of symmetry and the splitting of the communicating events, followed by quotient extraction. A procedure for generating a solution has been given, along with examples to demonstrate that the overall approach is sound and that the theory does indeed work. Application to a much larger example, which is significantly closer to industrial sized problems than previously encountered in the work by other researchers, is given in appendix C.

Comparison with other techniques has shown that our approach is able to solve problems which could not be realistically solved via those techniques and that this class of problems is extensive. On the other hand, our approach needs further work before we can satisfactorily solve the observational equivalence problem. An interesting assertion is that the graph theoretic approach with additional algorithms to solve observational equivalence problems is equivalent to the constructive algorithm with merge-split [144].

The strength of the graph theoretic approach is that issues such as deadlock, livelock, divergence, fairness, nondeterminism and strong connectedness present no difficulties for the theory since, from the graph theoretic viewpoint, the semantics of

the various labels are meaningless [4]. In fact any behavioural attribute of a machine can be ignored from the point of view of barriers to application. It is interesting to ask (and the subject of further research) which behavioural characteristics are invariant under quotient extraction [5].

The inherent reliance on symmetry proved to be a valuable validation tool, since non-compliance to certain symmetry constraints invariably indicated an error in the specification, usually incorrect transitions. This was an unexpected benefit.

Whether or not the algorithm can solve for completely disconnected machines needs further research, though we should stress the current algorithm is capable of solving problems based on disconnected $\mathcal{M}_p$ machines. The issue is under what conditions is it possible to solve for disconnected a $\mathcal{M}_r$.

The author also asserts that the graph theoretic approach is computationally less expensive than previous algorithms and will reduce protocol converter synthesis development times by up to an order of magnitude or more (see chapter 7). This statement is borne out by comparing the development times for the large example in the appendix C which includes estimates taken from a protocol engineering unit. An overview of processing times is given in figure 7.1, chapter 7.

The use of some elements of CCS has been non-standard in terms of not following Milner's strict definitions. The reason for this has been to highlight the need for extensions to CCS which encompass the essentially structural approach adopted here, as opposed to behaviour. The extensions of CCS, and the subsequent impact on the notation presented in this chapter, is discussed in chapter 6.

[4] While the discarding algorithm should also, in principle, be able to address such problems, it is not a practical proposition

[5] The author is currently preparing a paper which will pursue this issue further.

Chapter 6

Graph Theoretic Operators and Morphisms in CCS

Although CCS has a number of 'static' laws, we have showed that they are not rich enough in the context of graph theoretic applications to machine theory. This chapter defines a number of extensions to CCS that are essentially structural, or static in nature. This extended CCS (referred to as CCSg) provides the necessary syntax and semantics to connect the graph theoretic and bisimularity properties of machines.

6.1 Introduction

This chapter will propose a number of extensions to Milner's Calculus of Communicating Systems CCS [152], based on the research described in chapter 5. Previous solution strategies relied heavily on the behavioural aspects of the respective machines. Using these behavioural characteristics one is able to use Milner's CCS (as

with Parrow, Shields and Martin) without modification. The graph theoretic approach however, results in a theory that is generally structural, or topological, in nature — and such notions are not prevelant in CCS. For instance, bisimularity is central to CCS concepts of equivalence and bisimulation is a behavioural concept — it is concerned with what an external observer sees and not how the internal workings of the machine are structured. Moving the problem to the graph theoretic domain represents a fundamentally different approach to methods based on bisimulation syntax and semantics. In order to apply graph theoretic results to machine problems, where the machines are specified using CCS, we need not only to understand the behavioural characteristics, but also the structural properties. The general approach taken has been to redefine graph theoretic operators and morphisms into equivalent operators and equivalences which use the underlying syntax and semantics of CCS. For the purposes of this thesis, we will refer to this extended CCS as CCSg.

6.2 Rationale and definition

In this section we formally define the extensions to CCS. By placing the interface problem within the graph theoretic context it is perhaps not surprising that the emphasis, in terms of analysis, moves from the behavioural aspects of the relevant machines to their structural properties. Hence, concepts such as bisimulation (in particular observational equivalence) do not provide the characteristics required for what is essentially the analysis of the underlying machine graphs. In attempting to provide a more useful set of syntax and semantics the authors have developed a number of extensions to CCS which provide the structural attributes necessary.

So far, three new extensions have been defined. These are structural equivalence, structural composition and the rendezvous operator. As we will show, these ideas fit quite naturally within CCS and a number of formal relations between CCS and CCSg

will be presented.

6.2.1 Intrinsic and communicating τ-actions

It has emerged that it is useful to distinguish between those τ- actions that are performed by a machine, with those that are formed via communication with another machine. For example, consider two machines $\mathcal{M}_p$ and $\mathcal{M}_q$. $\mathcal{M}_p$ and $\mathcal{M}_q$ may each perform a number τ- actions, but may also communicate with each other via τ-actions. We want to distinguish between those τ-actions which are intrinsic to each machine, to those that form only via communication with another machine.

6.2.2 Definition

Let $\mathcal{M}_p$ and $\mathcal{M}_q$ be machines and further let $\tau \in \Lambda(p)$, $\alpha \in \Lambda(p)$ and $\overline{\alpha} \in \Lambda(q)$. Then the τ actions which are elements of $\mathcal{M}_p$ alone are called *intrinsic* τ-actions and denoted τ_i. If $\mathcal{M}_p$ and $\mathcal{M}_q$ are composed as $(\mathcal{M}_p \mid \mathcal{M}_q)\backslash A\alpha$, the α and $\overline{\alpha}$ actions are allowed to synchronise, resulting in a *communicating* τ-action, denoted τ_c.

6.2.3 Structural equivalence

Structural equivalence defines an isomorphism between machines, so that not only is behaviour indistinguishable, but also the underlying structure of the state transition graphs. The term 'equivalence' is used, as opposed to isomorphism, to show that this relation is to be viewed as part of the CCS equivalence hierarchy. It defines the strongest form of equivalence between two machines, as does isomorphism in the case of graphs.

6.2.4 Definition

$R^{\infty}(p)$ is defined to be the set of all states reachable from p via the set of all (possibly infinite) traces.

Definition 6.2.4 merely ensures that if p can reach another state, even if the sequence of transitions (trace) is possibly infinite, then that state must be included. We are now going to use this to define a special form of equivalence.

6.2.5 Definition

p_1 and q_1 are structurally equivalent, denoted $p_1 \stackrel{S}{=} q_1 \Leftrightarrow \exists$ bijective $\phi : R^{\infty}(p_1) \longrightarrow R^{\infty}(q_1)$ and $\forall p' \in R^{\infty}(p_1)$ and $\forall p'' \in R^{1}(p')$ and $\forall \alpha \in \Lambda(p')$ and $\forall q' \in R^{\infty}(q_1)$ and $\forall q'' \in R^{1}(q')$ and $\forall \beta \in \Lambda(q')$ then :-

$$\begin{aligned} \phi(p_1) &= q_1 \\ \phi(p') &\rightarrow^{\alpha} \phi(p'') \text{ if } p' \rightarrow^{\alpha} p'' \\ \phi^{-1}(q_1) &= p_1 \\ \phi^{-1}(q') &\rightarrow^{\beta} \phi^{-1}(q'') \text{ if } q' \rightarrow^{\beta} q'' \end{aligned}$$

Notice that the notion of bisimulation is not excluded here - we still expect the machines to be indistinguishable from an observers viewpoint. What we have done is to impose a further restriction on structure, so not only do the machines actually behave in the same way, but they also have the same structure. This extra condition is represented by the bijective mapping ϕ. In fact, structural equivalence is strong equivalence with the ϕ condition added. This leads to the following lemma.

6.2.6 Lemma

Structural equivalence, strong equivalence, observation congruence and observational equivalence form an equivalence hierarchy, with

$$\overset{S}{=} \Rightarrow \sim \Rightarrow = \Rightarrow \approx$$

Proof: That $\sim \Rightarrow = \Rightarrow \approx$ is a standard result from Milner's theory (proposition 7.2.4 [152]). Hence, we need only show that $\overset{S}{=} \Rightarrow \sim$. Let $p \overset{S}{=} q$, then by Definition 6.2.5 given $p \rightarrow^{\alpha} p' \; \exists q' : q \rightarrow^{\alpha} q'$ and $p' \overset{S}{=} q'$. Also, given $q \rightarrow^{\alpha} q' \; \exists p' : p \rightarrow^{\alpha} p'$ and $p' \overset{S}{=} q'$. But this is precisely the definition for strong equivalence in [152]. Hence, strong equivalence is a special case of structural equivalence where ϕ is the identity relation, which leads to the result $\overset{S}{=} \Rightarrow \sim$ and the proof is done. □

This notion of machine isomorphism also provides a link with topological equivalences. An example of a topological equivalence hierarchy is homeomorphism, homomorphism and isomorphism. The topological and machine hierarchies (via structural equivalence) meet at isomorphism, that is isomorphism is common to both topology and bisimularity.

Some properties of structural equivalence are now given.

$$\begin{aligned}
(\mathcal{M}_a \mid \mathcal{M}_b) \setminus A &\overset{S}{=} \mathcal{M}_a \setminus A \mid \mathcal{M}_b \setminus A \\
\mathcal{M}_a \setminus A &\overset{S}{=} \mathcal{M}_a \text{ if } A \cap \Lambda(\mathcal{M}_a) = \emptyset \\
\mathcal{M}_a \mid \mathcal{M}_b &\overset{S}{=} \mathcal{M}_b \mid \mathcal{M}_a \\
\mathcal{M}_a \mid (\mathcal{M}_b \mid \mathcal{M}_c) &\overset{S}{=} (\mathcal{M}_a \mid \mathcal{M}_b) \mid \mathcal{M}_c \\
\mathcal{M}_a \setminus A \setminus B &\overset{S}{=} \mathcal{M}_a \setminus (A \cap B) \\
(\mathcal{M}_a + \mathcal{M}_b)) \setminus A &\overset{S}{=} \mathcal{M}_a \setminus A + \mathcal{M}_b \setminus A \\
(\alpha.\mathcal{M}_a) \setminus A &\overset{S}{=} \begin{cases} \tau.\mathcal{M}_a & \text{if } \alpha \in A \\ \alpha.\mathcal{M}_a & \text{otherwise} \end{cases}
\end{aligned}$$

6.2.7 Structural composition

Structural composition is the machine equivalent to the graph Cartesian product. With this form of composition the machines are composed purely on a structural

basis, in accordance with the associated labeled digraphs. Unlike Milner's notion of composition, no merging of actions with their duals to perform τ_c actions is permitted. Hence the structure of the structurally composed machines is indistinguishable from the graph Cartesian product of their underlying labeled digraphs. As will be seen later, this concept is crucial for the application of graph theoretic concepts to the interface equation and quotient machine problems.

6.2.8 Definition

Let $\mathcal{M}_p$ and $\mathcal{M}_q$ be two machines with state sets $S(p), S(q)$ and action sets $\Lambda(p), \Lambda(q)$ respectively. Further let $S(p) \cap S(q) = \emptyset$. Then the structural composition of $\mathcal{M}_p$ and $\mathcal{M}_q$, denoted $\mathcal{M}_p \parallel \mathcal{M}_q$, is the machine with state set $S(p) \times S(q)$ such that two states (p_i, q_i) and (p_j, q_j) of $\mathcal{M}_p \parallel \mathcal{M}_q$ are adjacent if and only if either $p_i = p_j$ and there is a transition from q_i to q_j involving an action from $\Lambda(q)$ or $q_i = q_j$ and there is a transition from p_i to p_j involving an action from $\Lambda(p)$. It is both associative and commutative under structural equivalence.

6.2.9 Rendezvous operator and set

The rendezvous operator emerged as an alternative to Milner's hiding operator, the behaviour of which was not quite what we needed. The rendezvous operator endows the specifier with the option to disallow communication between actions which would otherwise be automatic using Milner's operator.

6.2.10 Definition

Let $\mathcal{M}_p$ and $\mathcal{M}_q$ be two machines with state sets $S(p), S(q)$ and action sets $\Lambda(p), \Lambda(q)$ respectively. Further let $\Lambda(p) \cap \Lambda(q) = \tau_i$. Let the machines be composed under structural composition to form the machine $\mathcal{M}_p \parallel \mathcal{M}_q$. Then the rendezvous operator

'$\wr$' and set A, which are incorporated into the interface equation as $(\mathcal{M}_p \parallel \mathcal{M}_q) \wr A$, are defined as follows. If $\alpha \in \Lambda(p)$ and $\overline{\alpha} \in \Lambda(q)$, but $\alpha \notin A$, then α and $\overline{\alpha}$ do not synchronise (or rendezvous using terminology from protocol engineering) to form a τ_c in $\mathcal{M}_p \parallel \mathcal{M}_q$. If $\alpha \in \Lambda(p)$ and $\overline{\alpha} \in \Lambda(q)$, but $\alpha \in A$ then α and $\overline{\alpha}$ rendezvous to form a τ_c in $\mathcal{M}_p \parallel \mathcal{M}_q$, with α and $\overline{\alpha}$ being hidden. If $\alpha \in A$, but with either $\alpha \in \Lambda(p)$ and $\overline{\alpha} \notin \Lambda(q)$ or $\alpha \notin \Lambda(p)$ and $\overline{\alpha} \in \Lambda(q)$, then, since no communication can take place, the rendezvous operator reverts to Milner's hiding operator and the action is hidden.

The last part of the definition is for completeness, our assumption is that the actions of the rendezvous set will always be those, and only those, that can communicate i.e. if $\alpha \in \Lambda(p)$ then $\overline{\alpha} \in \Lambda(q)$.

Given two structurally composed machines, the rendezvous operator and set provide the specifier with the means to control which actions are to communicate. We point out for interest that the algorithm developed by the authors works by reversing this process — it removes communication to recover the structurally composed machines via a τ-splitting transform [178].

6.2.11 Results

In this section we present a series of results which demonstrate the links between CCS and CCSg.

6.2.12 Theorem

Let $\mathcal{M}_p$ and $\mathcal{M}_q$ be machines and let $A = \{\alpha \mid \alpha \in \Lambda(p) \text{ and } \overline{\alpha} \in \Lambda(q)\}$, further let $A \neq \emptyset$, then $(\mathcal{M}_p \parallel \mathcal{M}_q) \wr A \stackrel{S}{=} (\mathcal{M}_p \mid \mathcal{M}_q) \backslash A$

Proof: The strategy for the proof is to show that whatever $(\mathcal{M}_p \parallel \mathcal{M}_q) \wr A$ can do $(\mathcal{M}_p \mid \mathcal{M}_q) \backslash A$ does the same, and vice versa. Let $s_1 = (p_1, q_1)$ be a state of

$(\mathcal{M}_p \parallel \mathcal{M}_q) \wr A$ and let $\Lambda(s_1)$ be the action set of s_1. Let $\mu \in \Lambda(s_1) - A$, then by Definition 6.2.8 μ is associated with either a p_1 or a q_1 transition. If $\mu \in A$, then by Definition 6.2.10 s_1 performs a τ_c and $\mu \notin \Lambda(s_1)$. Clearly s_1 is a state of $(\mathcal{M}_p \mid \mathcal{M}_q)\backslash A$, by Definition of Milner's composition operator. Further, if $\mu \in \Lambda(s_1) - A$, then by definition of Milner's composition operator μ must be associated with either a p_1 or a q_1 transition. If $\mu \in A$, then by definition of Milner's hiding s_1 performs a τ_c and $\mu \notin \Lambda(s_1)$. Hence if $s_1, \Lambda(s_1)$ is in $(\mathcal{M}_p \parallel \mathcal{M}_q) \wr A$ then $s_1, \Lambda(s_1)$ is in $(\mathcal{M}_p \mid \mathcal{M}_q)\backslash A$, hence $(\mathcal{M}_p \parallel \mathcal{M}_q) \wr A \subseteq (\mathcal{M}_p \mid \mathcal{M}_q)\backslash A$. It is a simple matter to show that the converse is also true, i.e. $(\mathcal{M}_p \parallel \mathcal{M}_q) \wr A \supseteq (\mathcal{M}_p \mid \mathcal{M}_q)\backslash A$. Hence by Definition 6.2.5 $(\mathcal{M}_p \parallel \mathcal{M}_q) \wr A \stackrel{S}{=} (\mathcal{M}_p \mid \mathcal{M}_q)\backslash A$ and the proof is done.

□

This theorem shows that if $\mathcal{M}_p$ and $\mathcal{M}_q$ have the ability to communicate, and that all the relevant actions are included within A, then the machine resulting from Milner's standard CCS notation is structurally equivalent to that obtained via CCSg. In the case where no communication takes place between $\mathcal{M}_p$ and $\mathcal{M}_q$, i.e. $\forall \alpha \in \Lambda(p)\ \not\exists \overline{\alpha} \in \Lambda(q)$, then the rendezvous operator defaults to Milners hiding operator.

6.2.13 Lemma

Let A be defined as in theorem 6.2.12 and let $B \subseteq \Lambda(p) - A$ and $\overline{B} \cap \Lambda(q) = \emptyset$, or $B \subseteq \Lambda(q) - A$ and $\overline{B} \cap \Lambda(p) = \emptyset$ and $(A \cup \overline{A}) \cap (B \cup \overline{B}) = \emptyset$, then $(\mathcal{M}_p \parallel \mathcal{M}_q) \wr A\backslash B \stackrel{S}{=} (\mathcal{M}_p \mid \mathcal{M}_q)\backslash A \cup B$.

Proof: By theorem 6.2.12 $(\mathcal{M}_p \parallel \mathcal{M}_q) \wr A \stackrel{S}{=} (\mathcal{M}_p \mid \mathcal{M}_q)\backslash A$. By Definition 6.2.10 if $\alpha \in B$ and hence not involved in communication, then '$\wr\{\alpha\}$' reverts to '$\backslash\{\alpha\}$'. This implies that $(\mathcal{M}_p \parallel \mathcal{M}_q) \wr A\backslash B \stackrel{S}{=} (\mathcal{M}_p \mid \mathcal{M}_q)\backslash A\backslash B$. By proposition 4.3.8.5 in [152] $(\mathcal{M}_p \mid \mathcal{M}_q)\backslash A\backslash B \sim (\mathcal{M}_p \mid \mathcal{M}_q)\backslash A \cup B$. Since $B \subseteq \Lambda(p) - A$ and $\stackrel{S}{=} \cap \Lambda(q) = \emptyset$, or $B \subseteq \Lambda(q) - A$ and $\stackrel{S}{=} \cap \Lambda(p) = \emptyset$ and $(A \cup \overline{A}) \cap (B \cup \overline{B}) = \emptyset$, $(\mathcal{M}_p \mid \mathcal{M}_q)\backslash A\backslash B \stackrel{S}{=}$

$(\mathcal{M}_p \mid \mathcal{M}_q)\backslash A \cup B \Rightarrow (\mathcal{M}_p \parallel \mathcal{M}_q) \wr A\backslash B \stackrel{S}{=} (\mathcal{M}_p \mid \mathcal{M}_q)\backslash A \cup B$. QED.

□

6.2.14 Proposition

Let $(\mathcal{M}_p \parallel \mathcal{M}_q) \wr A \stackrel{S}{=} (\mathcal{M}_p \mid \mathcal{M}_q)\backslash A$, where A is defined as in theorem 6.2.12. Define machines $\mathcal{M}'_p$ and $\mathcal{M}'_q$ such that $\mathcal{M}'_p$ and $\mathcal{M}'_q$ are bisimulations of $\mathcal{M}_p$ and $\mathcal{M}_q$ respectively. Then $(\mathcal{M}'_p \parallel \mathcal{M}'_q) \wr A \stackrel{S}{=} (\mathcal{M}'_p \mid \mathcal{M}'_q)\backslash A$.

Proof: Trivial. Clearly $\mathcal{M}'_p$ and $\mathcal{M}'_q$ are well defined machines. Since theorem 6.2.12 holds for all machines (subject to the conditions on the set A) proposition 6.2.14 must follow. □

6.2.15 Proposition

Given $(\mathcal{M}_p \parallel \mathcal{M}_r) \wr A \stackrel{S}{=} \mathcal{M}_q$ and $B \subseteq \Lambda(p) - A$ and $B \cup \overline{B} \cap \Lambda(r) = \emptyset$ then $\mathcal{M}_q\backslash B \stackrel{S}{=} (\mathcal{M}_p \parallel \mathcal{M}_r) \wr A\backslash B \stackrel{S}{=} (\mathcal{M}_p\backslash B \parallel \mathcal{M}_r) \wr A$. Similarly, if $B \subseteq \Lambda(r) - A$ and $B \cup \overline{B} \cap \Lambda(p) = \emptyset$ then $\mathcal{M}_q\backslash B \stackrel{S}{=} (\mathcal{M}_p \parallel \mathcal{M}_r\backslash B) \wr A$.

Proof: By proposition 4.3.8.7 [152] $(\mathcal{M}_p \mid \mathcal{M}_q)\backslash B \sim (\mathcal{M}_p\backslash B \mid \mathcal{M}_q\backslash B)$. Since B contains only those actions of $\mathcal{M}_p$ (or $\mathcal{M}_q$) which are not involved in communication, it follows from proposition 4.3.8.4 [152] that $\mathcal{M}_q\backslash B \sim \mathcal{M}_q \Rightarrow (\mathcal{M}_p \mid \mathcal{M}_q)\backslash A\backslash B \sim (\mathcal{M}_p\backslash B \mid \mathcal{M}_q\backslash B)\backslash A \Rightarrow (\mathcal{M}_p \mid \mathcal{M}_q)\backslash A\backslash B \sim (\mathcal{M}_p\backslash B \mid \mathcal{M}_q)\backslash A \Rightarrow$ by Theorem 6.2.12 $\stackrel{S}{=} (\mathcal{M}_p\backslash B \mid \mathcal{M}_q) \wr A$. The same argument applies when B contains actions from $\mathcal{M}_q$. QED.

□

6.3 Application

This section shows the application of these new concepts to the interface equation, which in turn can be used to generate protocol gateway (converter) specifications. The

treatment given is necessarily simple and brief. The details of the graph theoretic solution process can be found in [178]. An interface equation, using the notation described, is expressed as :

$$(\mathcal{M}_p \mid \mathcal{M}_r)\backslash C \stackrel{S}{=} \mathcal{M}_q, C = A \cup B \text{ where } A = \{\alpha : \alpha \in \Lambda(p) \wedge \overline{\alpha} \in \Lambda(r)\} \text{ and}$$
$$B \subseteq \Lambda(p) \cup \Lambda(r) - A.$$

Where $\mathcal{M}_p$ and $\mathcal{M}_q$ are known machines and the interface machine $\mathcal{M}_r$ needs to be found. Let $\mathcal{M}_p$ and $\mathcal{M}_q$ be defined in CCS prefix notation as follows:-

$$\begin{aligned} p_1 &\Leftarrow \alpha p_3 + \alpha p_2 \\ p_2 &\Leftarrow \gamma p_3 \\ p_3 &\Leftarrow \tau_i p_1 \end{aligned}$$

$$\begin{aligned} q_1 &\Leftarrow \delta q_4 \\ q_2 &\Leftarrow \gamma q_3 + \delta q_5 \\ q_3 &\Leftarrow \delta q_6 + \tau q_1 \\ q_4 &\Leftarrow \tau q_2 + \tau q_3 \\ q_5 &\Leftarrow \gamma q_6 \\ q_6 &\Leftarrow \tau q_4 \end{aligned}$$

We need to find the machine $\mathcal{M}_r$ which when structurally composed with $\mathcal{M}_p$ and with the appropriate actions in the rendezvous set, is structurally equivalent to $\mathcal{M}_q$.

Via an algorithm that uses symmetry and set-theoretic arguments the τ actions are labeled as either τ_i or τ_c actions. There are only two τ_c actions, namely $q_4 \Leftarrow \tau q_2 + \tau q_3$.

The next step is to establish the embedded components of the $\mathcal{M}_p$ machine within $\mathcal{M}_q$. In the case where $\mathcal{M}_p$ is reachable this is done by removing the actions of $\mathcal{M}_q$ not done by $\mathcal{M}_p$. This will leave a machine $\mathcal{M}'_q$ which has the embedded components of $\mathcal{M}_p$. Each embedded $\mathcal{M}_p$ corresponds to a state of the $\mathcal{M}_r$ machine.

Notice that we can use Theorem 6.2.12 and Proposition 6.2.15 to rewrite this equation as :-

$$(\mathcal{M}_p \backslash B \parallel \mathcal{M}_r) \wr A \stackrel{S}{=} \mathcal{M}_q$$

That is, all the actions performed by $\mathcal{M}_p$ but which are not involved in communication can be removed, since they have no affect the structure (or behaviour) of $\mathcal{M}_r$.

$$\begin{aligned} q_1 &\Leftarrow \mathbf{0} \\ q_2 &\Leftarrow \gamma q_3 \\ q_3 &\Leftarrow \tau_i q_1 \\ q_4 &\Leftarrow \mathbf{0} \\ q_5 &\Leftarrow \gamma q_6 \\ q_6 &\Leftarrow \tau_i q_4 \end{aligned}$$

Using symmetry arguments one can easily establish that the two embeddings of $\mathcal{M}_p$ in $\mathcal{M}_q$ are given by $\{q_1, q_2, q_3\}$ and $\{q_4, q_5, q_6\}$, with the following mapping (which is unique up to isomorphism) :-

$$\begin{aligned} q_1 &\mapsto (p_1 \parallel r_1) \\ q_2 &\mapsto (p_2 \parallel r_1) \\ q_3 &\mapsto (p_3 \parallel r_1) \\ q_4 &\mapsto (p_1 \parallel r_2) \\ q_5 &\mapsto (p_2 \parallel r_2) \\ q_6 &\mapsto (p_3 \parallel r_2) \end{aligned}$$

The next step is to identify the actions that the original $\mathcal{M}_p$ did, but which the embedded components do not. These are identified with the τ_c actions, which are then 'split' to recover the original actions, using the τ-splitting transform Υ. The

splitting process can be viewed as a projection of a τ_c-transition in the $\mathcal{M}_q$ machine onto the $\mathcal{M}_p$ and $\mathcal{M}_r$ machines (which are the underlying components of $\mathcal{M}_q$). Let $q_i \xrightarrow{\tau_c} q_j$, then $\Upsilon(q_i \xrightarrow{\tau_c} q_j) \mapsto (q_k \rightarrow^{\alpha} q_l, q_m \rightarrow^{\overline{\alpha}} q_n)$. In the example above the original $\mathcal{M}_p$ machine did $p_1 \Leftarrow \alpha p_3 + \alpha p_2$, which the embedded components do not do. What is more, the sense of the transitions is mirrored in the τ_c transitions, but these also include a state change between the embedded components, in other words $\mathcal{M}_r$ transitions. Thus $\Upsilon(q_4 \xrightarrow{\tau_c} q_2) \mapsto (q_4 \rightarrow^{\alpha} q_5, q_4 \rightarrow^{\overline{\alpha}} q_1)$ and $\Upsilon(q_4 \xrightarrow{\tau_c} q_3) \mapsto (q_4 \rightarrow^{\alpha} q_1, q_4 \rightarrow^{\overline{\alpha}} q_6)$. Thus we rebuild the $\mathcal{M}_p$ machine within $\mathcal{M}_q$ by recovering the actions that $\mathcal{M}_p$ donated to the communication process, until all communicating τ_c actions have been removed and we have $(\mathcal{M}_p \parallel \mathcal{M}_r) \stackrel{S}{=} \mathcal{M}'_q$, where $\mathcal{M}'_q$ is the restructured $\mathcal{M}_q$ machine after the splitting process. It is at this stage that graph quotient algorithms are applied in order to establish the structure of the unknown $\mathcal{M}_r$. In fact we assert that $\mathcal{M}_r = \frac{\mathcal{M}_q}{\mathcal{M}_p}$. In terms of the processing, we need consider only those actions performed by $\mathcal{M}_q$ but not by $\mathcal{M}_p$, i.e. $\Lambda(q) - \Lambda(p)$, since the $\mathcal{M}_r$ machine has no common actions (other than τs). The specification of the $\mathcal{M}_r$ machine is trivial in this case and is defined as :-

$$_1 \Leftarrow \delta r_2$$

$$r_2 \Leftarrow \overline{\alpha} r_1$$

A number of complexities have been ignored in this example, here the aim has been to demonstrate that the extensions to CCS proposed here do indeed have a sound basis.

6.4 Summary

In this chapter the author has presented extensions to CCS, which emerged from research into graph theoretic techniques for solving CCS interface equations. The extensions proposed provide CCS with additional syntax and semantics by defining

what is essentially an overlap between CCS and graph theory. By doing this we not only analyse the behaviour of a machine, but also its structure. We have shown that these structural concepts can be included quite naturally within standard CCS. On the theoretical side, the move towards graph theory provides a whole new set of possibilities in terms of the topological analysis of machines.

Chapter 7

Conclusions

This chapter summarises and critiques the findings of the research. The initial set of objectives and the results obtained with respect to these are discussed along with a discussion of the strengths and shortcomings of the research presented here.

7.1 Summary of thesis

This thesis has presented a novel technique for solving the interface equation in the context of protocol converter synthesis. Protocol conversion is likely to be a pressing problem for the communications community for the foreseeable future. Indeed, with the emergence of new and advanced services and the information superhighway [84, 55, 87], the problem of protocol conversion is likely to increase exponentially in seriousness.

The application of graph theory proved to be decisive in terms of approaching the research objectives on a broad front. While not a key objective, it was hoped that the graph based approach would prove to be completely general. In hindsight, this was rather unrealistic : the connection between topology and bisimulation was more

problematic than first thought.

Empirical analysis has shown that the approach is a significant improvement on previous methods. While not completely general, it significantly widens the class of solvable problems significantly.

7.2 Results

The research described in this thesis set out with six initial aims (see chapter 4). These are given below along with the results achieved.

Objective To address and solve the computational complexity issues associated with other methods. All previous algorithms suffered from being intractable [94] in some cases. Indeed some authors felt that the automatic construction of protocol converters was impossible in polynomial time [125]. Current industrial methods are labour intensive, with a 2000 state problem taking anything up to 3 man-years to develop. Protocol converters are usually developed when a communications company wants to introduce a new service; hence the sooner the converter is developed and installed, the sooner the associated service can be offered to customers and the sooner it can start generating revenue. Hence the gains are at both ends — reduced development costs and increased revenue generation. Thus the importance of this issue.

Result Achieved. Figure 7.1 provides some indication as to the expected gains made in this area. While the data is crude, it is clear that with suitable automation the algorithm is at least an order of magnitude the better with respect to processing time than previous approaches. Whereas the algorithms of previous researchers involved exhaustive searching, our approach is focussed at the key computational problem, namely dealing with the communicating τ actions. By way of

a comparison, it would have taken the discarding algorithm $\mathcal{O}(2^{n_p n_q}) \Rightarrow 2^{24220}$ days to solve the example given in appendix C automatically. The constructive algorithm would have improved this, but still be $\mathcal{O}(n_p^3 n_q^3 \mu^{n_p})$ where μ is a measure of the duplication of q states and ranges from 1 to n_q(see chapter 4). An estimate for the constructive algorithm for solving this example automatically was 118 days ($\mu = n_q/10$). The graph algorithm complexity is $\mathcal{O}(n_p^3 n_q^4 \mid \tau \mid^2)$, where $\mid \tau \mid$ is the number of τ actions which, using the same state per second processing rate of the constructive algorithm, gives 5.2 hours for automatic processing. The example in appendix C took the author 6 hours of manual effort. Interestingly, a protocol engineer estimated that the example in question would have taken of the order of 30–40 mandays to solve using a standard semi-manual approach.

These figures are of course crude, but they do give some idea of the processing gains achievable with the graph-theoretic approach and illustrates the non-viability of the discarding algorithm.

Objective Since previous algorithms are based on a behavioural approach, behavioural characteristics such as livelock and deadlock can have a detrimental effect on the algorithms. The effect deadlock and livelock have on the constructive algorithm, for example, is identical to that of non-A reachability and a solution will not be found. By adopting a structural approach, it was felt that such behavioural characteristics could be effectively ignored.

Result Achieved. By adopting an almost exclusively structural approach all behavioural considerations were indeed irrelevant in terms of the effectiveness of the algorithm. Although it is asserted that the discarding algorithm is able to address such problems, in terms of efficiency it is not an option.

Objective To overcome the non-A reachability limitation of Martin and Shield's work. In short, this required the $\mathcal{M}_p$ to be connected with respect to non-A actions. This condition prevented $\mathcal{M}_p$ becoming disconnected which, for a path searching algorithm, is clearly a problem.

Result Achieved. While we have found pathological machine systems which are unsolvable (typically highly degenerate systems), all other problems which contained disconnected $\mathcal{M}_p$ machines were solved. This was achieved by the construction of the $\tilde{\Pi}$-planes.

Objective To develop a technique which is able to identify quickly whether a solution is possible for the majority of cases.

Result Achieved. One of the surprises of the research was the power of using symmetry arguments to both validate the relevant specifications and to determine quickly whether a solution would be possible. If full symmetry cannot be achieved (with respect to the GCP), then either the specifications are incorrect or there does not exist any $\mathcal{M}_r$ machine which can be accommodated within the overall structure which, in turn, infers the lack of a solution. This is apparent at Step 2 of the algorithm (chapter 5).

Objective Attempt to address scalability. This is a key criticism of the work of many other researchers. The objective was to find suitable industrial examples to which the theory would be applied.

Result Partially achieved. While the example given in appendix C was larger by at least 1.5 orders of magnitude than previously published work, the hope was to apply the theory to real, industrial examples, such as DPNSS and DASS2 (at least 2000 states). Unfortunately these problems were specified in English and the effort involved in converting them to CCS was considered prohibitive. Some

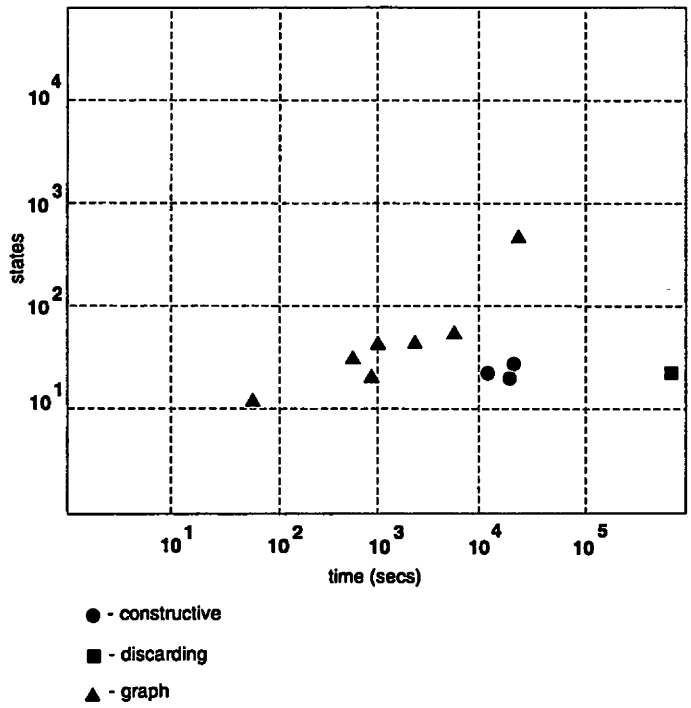


Figure 7.1: Processing capabilities of a selected range of algorithms

protocols were available in LOTOS, but these were not either large enough or complete. Hence, the alternative option — to produce an artificial example that nevertheless would include many problematic characteristics that would be encountered in real protocols.

Objective To broaden the class of solvable problems.

Result Achieved. The class of problem that can now be addressed includes all forms of nondeterminism, anomalous behaviour and disconnected $\mathcal{M}_p$ machines. Table 3 shows how the work described here replaces some methods and complements other methods. Thus, the protocol engineer has a potentially powerful toolkit whereby virtually any protocol synthesis problem can be addressed automatically.

7.3 Conclusions

Prior to this research the range of solvable problems could be summarised as indicated in Table 2. The most general method was the work of Norris, Shields and Martin (DA2), however this algorithm has no practical value as was demonstrated in chapter 4. We include it here for completeness.

The following tables summarise the the effectiveness of various algorithms on a number of interface equations with particular characteristics — both prior to the research (Table 2) and after (Table 3). Explanations of the abbreviations are given in Table 1.

Table 1 - Key	
WD	weakly determinate
D	determinate
ND	nondeterminate
DA1	weakly determinate discarding algorithm
DA2	general discarding algorithm
CA1	constructive algorithm
CA2	constructive algorithm with merge split
GA	graph based algorithm
Tab	Parrow's tableaux method
O	observational equivalent
S	strongly equivalent
St	structurally equivalent

Table 2		
Interface Equation	Properties	Optimum solution methods (increasing cost left to right)
$(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \approx \mathcal{M}_q$	WD $\mathcal{M}_q$, O	CA1, CA2, DA1, DA2
$(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \approx \mathcal{M}_q$	D $\mathcal{M}_q$, O	Tab, CA1/2, DA1/2
$(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \sim \mathcal{M}_q$	ND $\mathcal{M}_q$, S	CA1/2, DA2

The situation is now as follows, with the domain of application significantly expanded.

Table 2		
Interface Equation	Properties	Optimum solution methods
$(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \approx \mathcal{M}_q$	WD $\mathcal{M}_q$, O	CA1, CA2, DA1, DA2
$(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \approx \mathcal{M}_q$	D $\mathcal{M}_q$, O	Tab, CA1/2, DA1/2
$(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \sim \mathcal{M}_q$	ND $\mathcal{M}_q$, S	GA, CA1, CA2, DA2
$(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \stackrel{S}{=} \mathcal{M}_q$	ND $\mathcal{M}_q$, St	GA, CA1, CA2, DA2
$(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \stackrel{S}{=} \mathcal{M}_q$	M's disconnected, St	GA, CA2, DA2
$(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \sim \mathcal{M}_q$	M's are deadlocked etc	GA

The following emerge from these tables :-

1. Theoretically, the discarding algorithm is still the most general — but in terms implementation a state explosion quickly ensues for all but the most trivial of examples. Hence, it is not a practical proposition.

2. Prior to this research, the constructive algorithm was the most efficient method.

3. When applicable, the graph theoretic algorithm is by far the most efficient.

4. The graph theoretic algorithm extends the class of solvable problems.

5. For machines with anomalous behavioural characteristics, only the graph theoretic algorithm can be used.

In terms of the original objectives, the research has demonstrated that all of them can be dealt with a single step — namely transferring the problem to the graph theoretic domain. In undertaking this research a number of strengths and weaknesses, or issues, have become apparent. Firstly, the gains made have been achieved at a price, namely the theory cannot realistically cope with observational equivalence. Were it able to, then the research described here would constitute the nearest that has been

developed to a general calculus of conversion. Indeed, it would be equivalent to Martin's constructive algorithm with merge-split. It is an open question as to which would prove the most efficient. Secondly, it was with some disappointment that we were unable to identify a suitable industrial strength example. In terms of future research this problem must have a high priority.

There is also an interesting scenario where the graph algorithm would struggle to find a solution and that is with a problem which contained perfect symmetry. It is ironic that an algorithm based on symmetry should also be limited by it. This problem is intimately connected with the entropy idea discussed in chapters 5. Having said that, both the author and an experienced colleague with substantial expertise in CCS, were unable to produce such an example and we doubt whether it is actually feasible. Nevertheless this is an interesting result and provides, in conjunction with the observational equivalence problem, some idea of the current limitations of the graph theoretic approach.

In order to utilise this work in an industrial setting, a coordinated approach must be adopted. Attempts to produce a tool in isolation and then to expect protocol engineers to use it, is a trap that many technology transfer projects have fallen into in the past. It has to be understood that this is the first step and that a number of further steps need to be taken before this work can be packaged in the appropriate manner. The aim, then should be to produce a methodology covering the end-to-end protocol development process, for which the graph theoretic approach, supplemented by other techniques, would form the core.

On balance then, the research has been very successful in addressing the key issues encountered by previous researchers in the field of formal protocol converter synthesis.

Chapter 8

Future research

This chapter identifies a number of research themes which could form the basis for future work. They are split into two categories, one being theoretical, the other practical. In some cases the author has provided initial thoughts as to how the particular research problem might be pursued.

8.1 Introduction

This thesis has bridged a number of topics; these are protocol engineering, computer science, software engineering and graph theory. Apart from solving the initial problem statement, the convergence of the concepts and ideas from these various domains has acted as a catalyst in identifying a number of potentially rich fields of study. This chapter summarises some of the more tractable problems that were identified.

There are two types of work : that which bridges the gap between research and industry and that which is theoretical.

The theoretical work adopts a broader perspective and looks beyond the protocol

problem to issues such as a general calculus of conversion, quotient invariants and the link between structure and behaviour.

8.2 Practical

8.2.1 Language limitations of CCS

CCS does not figure large in any strategic initiatives within the telecommunications industry. While CCS has provided an ideal vehicle as a research tool, it has severe limitations concerning applied development. An alternative variant of the theory, based on notations such as SDL and LOTOS is preferable. It is the authors view that, given its relationship with CCS, initial attention should be directed at LOTOS. LOTOS has been used in the telecommunications industry as the specification language for OSI protocols. It has also been used to specify complex systems. Initial studies suggest that the differences between LOTOS and CCS are largely syntactic and that mapping LOTOS onto directed graphs should prove no more difficult than for CCS.

8.2.2 Real protocols and scalability

The best means of convincing industry that a particular method is ripe for industrial exploitation is to apply it to problems which industry understands and believes in. This can only be achieved by industrial-strength examples. We also need to determine which class of industrial problems can be solved; and if they can't, address the reasons why not. The application of the quotient graph algorithm to real protocols is clearly a major step. Before this is done other issues such as the use of alternative notations would need to be addressed first.

8.2.3 Novel hardware and software architectures

What can dedicated hardware implementations and novel architectures based on the algorithms offer? Can systolic arrays [123], or neural nets [96] , for instance, provide a quantum leap in terms of processing times? Given that much of the algorithm can be expressed in terms of matrices; the use of digital array processors and systolic array processors could prove a fruitful avenue. As well as the technical issues, the cost/benefits of this approach would need to be investigated, since dedicated silicon based solutions could be costly to produce. It would also be interesting to examine whether the solution algorithm could be improved using a parallel algorithm. The use of such parallel algorithms for graph analysis has been investigated by [209], where attempts were made to parallelise certain partitioning algorithms without success. It would appear that the nature of many graph algorithms makes parallelism hard to achieve. The algorithm presented in this thesis would be difficult to parallelise due to the interdependency of some of the sub-algorithms (τ-splitting and symmetric closure for example) and the cause and effect relationships between one step and the next. It is likely that any parallel implementation of the graph–theoretic algorithm will be non-tr ivial.

8.2.4 Tool support

Without appropriate tools, the method described here will have little chance of success. The development of an appropriate toolset is therefore paramount. A potential problem — and hence a challenge for the researcher — would be a graphical user interface capable of presenting very large graphical structures in a meaningful way. One approach might be a windows-based application which provides the facility for looking at various parts of the graph structure, but which are indexed and linked in some way to aid navigation. For the protocol engineer such a facility would be

invaluable, since it is the nature of the task to examine possible links between various structures [122].

8.3 Theoretical

8.3.1 Observational equivalence

Observational equivalence — if it were to become a practical issue — presents the most difficult problem for the graph theoretic approach. In basic terms, too much information is lost during the reduction process and the symmetry constraints which are sufficient to reconstitute the composite graph via symmetric closure, are no longer valid. This may form part of a more general look at machine morphisms, as discussed by Warner in [242]. A possible approach to this problem is presented in appendix D.

8.3.2 Developing a generalised interface theory

Is the quotient approach developed in this thesis a generalisation of all protocol conversion algorithms? Can it therefore form the basis of a general theory of interfaces? In an interesting recent paper Lam and Shanker [127] propose a theory of module interfaces (as opposed to a general theory) uses graph theory as a basis. It is the view of the author that graph theory could well provide an ideal medium for the generation of such a general theory.

8.3.3 Partitioning

Partitioning could significantly reduce the state and action space. Hence it is an interesting technique which could prove useful for solving large-scale protocol conversion problems. The interface equation algorithm could then be used to solve the partitioned machines, thus reducing the computational time of the problem. We also need to consider the problem of displaying these large complex structures on support

tools (as discussed above). Algorithms have been developed which can partition large graph structures in polynomial time [237, 46, 252, 189, 20, 235, 208, 77, 2, 8]. Such an approach was considered early in the author's research. The main problem with partitioning, it was found, was refining the (reduced) solution to a sufficient level of detail from which the fine structure of the protocol could be determined. The reason for this is that the partitioning process usually results in a loss of information.

8.3.4 CCSg

The extensions to CCS presented in this thesis arose from the need to analyse the structural properties of machines as well as their dynamic (behavioural) characteristics. Chapter 6 presented some elementary results and showed that CCSg sits quite naturally within the CCS domain. However, the author feels that there is scope for further research not only in terms of developing CCSg further, e.g. how it might be used in a practical setting, but also by exploring the deeper relationships between CCSg and standard CCS. This could be linked with the research proposed in section 8.3.8.

8.3.5 Merge-split

Whilst not directly relevant to the graph theoretic approach, the development of a merge-split algorithm for Martin's constructive approach could prove very rewarding. The anticipation is that the constructive algorithm with merge-split would be as general as the graph theoretic approach with some means for addressing observational equivalence. We would thus have three completely general methods of solving the interface equation (including the discarding algorithm). The question of primary interest is which would be the most efficient.

8.3.6 Iterative methods

This research theme is closely related to the observational equivalence research proposal. This research would look at possible iterative methods for solving the interface equation. Martin's constructive algorithm could provide a starting point for this work. This is a fascinating area for research, where convergent and divergent behaviour could be studied. A possible starting point is given in appendix E.

8.3.7 Invariants

A given machine has a host of properties associated with its dynamic behaviour — deadlock, livelock, divergence or convergence, and fairness but to name a few. The object of future research in this area will be to establish whether these properties are preserved or are invariant under the quotient operation. So, given the interface equation $(\mathcal{M}_p \mid \mathcal{M}_r) \wr A \backslash B \approx \mathcal{M}_q$ and if $\mathcal{M}_p$ and $\mathcal{M}_q$ are both fair [78] for example, would $\mathcal{M}_r$ be fair? The questions are :-

1. which of these can properly be called general invariants,
2. which are invariants under special preconditions,
3. which are generally not invariants at all?

This work would have implications for the verification and validation of the quotient machines, since invariant properties need not be explicitly proved.

8.3.8 Structure and behaviour

The link between quotient machines and quotient graphs is suggestive of deeper links between machine behaviour and topology. Similar observations have been made by other authors such as Arbib [3]. Inductive inference [59] has been applied to the problem of deriving structure on the basis of behaviour [1]. Burks [27], has examined the

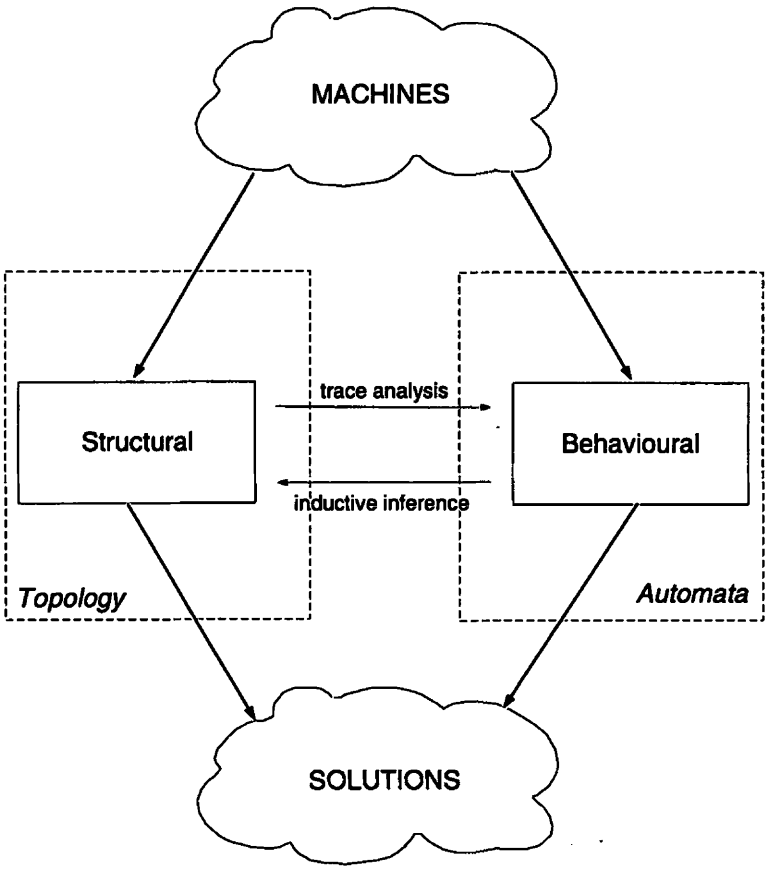


Figure 8.1: Relating topology and machines.

relationship between behaviour and structure for fixed and growing automata. Reggia and Codd [191, 45] have demonstrated that if the cell states of cellular automaton meet certain symmetry requirements, then self-replication behaviour is greatly enhanced. A flowchart, which positions the authors research in the context of generalised automata theory, can be found in appendix F. This research would examine the inter-relationships shown in figure 8.1. Is there a deep connection between the static and dynamic (behavioural) properties of machines. The diagram suggests that trace analysis and inductive inference are indeed examples of such a connection.

Bibliography

[1] D. Angluin and C. Smith. Inductive inference : Theory and methods. *Computing Surveys*, 15:3, 1983.

[2] C. Arbib. A polynomial characterisation of some graph partitioning problems. *Information Processing Letters*, 26(5), 1988.

[3] M. Arbib and E. Manes. Adjoint machines state–behaviour machines and duality. *Journal of Pure and Applied Algebras*, 6, 1975.

[4] M. Arbib and E. Manes. A category–theoretic approach to systems in a fuzzy world. *Journal of Pure and Applied Algebras*, 6, 1975.

[5] C. Ashworth and M. Goodland. *SSADM : A Practical Approach.* McGraw–Hill, 1990.

[6] J. Auerbach. TACT: a protocol conversion toolkit. *IEEE Journal on Selected Areas in Communications*, 8(1), 1990.

[7] F. Aurenhammer, J. Hagauer, and W. Imrich. Cartesian graph factorisation at logarithmic cost per edge. *Computational Complexity*, 2:331–349, 1992.

[8] E. Barnes. Partitioning large graphs. In *Proc. 1982 IEEE Int. Large Scale Systems Symp.* IEEE, 1982.

[9] M. Behzad and G. Chartrand. *Introduction to the Theory of Graphs.* Allyn and Bacon, 1971.

[10] H. Bekic. *Programming Languages and their Definition.* Springer–Verlag, 1985.

[11] M. Benyon. Protocol specification in concurrent systems software development. Technical report, Dept. Computer Science, University of Warwick, 1990.

[12] J. Bergstra and J. Klop. Algebra for communicating processes with abstraction. *Journal of Theoretical Computer Science*, 37:77–121, 1985.

[13] L. Von Bertelanffy. *General Systems Theory.* Penguin books, 1981.

[14] E. Best. Sequential and concurrent behaviour in Petri net theory. *Theoretical Computer Science*, 55, 1987.

[15] J. Billington. A high–level Petri net specification of the cambridge fast ring m–access service. In *Specification and Verification of Concurrent Systems.* Spriner–Verlag, 1990.

[16] G.V. Bochman and P. Mondain-Monval. Design principles for communication gateways. *IEEE Trans. on Selected Areas in Comms.*, 8(1), 1990.

[17] G. Bochmann and C. Sunshine. Formal methods in communication protocol design. *IEEE Trans. on Comms.*, 28(4), 1990.

[18] G. Boole. An investigation of the laws of thought, 1849.

[19] R. Booth. Using LOTOS for OSI. In *Colloquium on Formal Methods for Protocols.* IEE, 1991. Digest No. 1991/043.

[20] A. Bouloutas and P. Gopal. Some graph partitioning problems and algorithms related to routing in large computer networks. In *IEEE 9th Int. Conf. on Distributed Computing Systems.* IEEE, 1989. 89CH2706–0.

[21] L. Bourguet. A Petri Net tool for service validation in protocols. In *Protocol Specification Testing and Verification VI.* ,North–Holland, 1987. Proc. IFIP WG 6.1.

[22] J. Bowen. A brief history of algebra and computing. *IMA Bulletin*, 31(1):6–9, 1995.

[23] E. Brinksma. A tutorial on LOTOS. In *Protocol Specification Testing and Verification V.* ,North–Holland, 1986. Proc. IFIP WG 6.1.

[24] E. Brinksma. A theory for the derivation of tests. In *Proc. IFIP WG6.1 8th Int. Symp. on Protocol Spec. Testing and Ver.* ,North–Holland, 1988.

[25] E. Brinksma, G. Scollo, and C. Steenbergen. LOTOS specifications their implementation and their tests. In *Protocol Specification Testing and Verification VI.* ,North–Holland, 1987. Proc. IFIP WG 6.1.

[26] S. Brookes and A. Roscoe. Deadlock analysis in networks of communicating processes. In *Logics and Models for Concurrent Systems.* NATO ASI Series, 1985.

[27] A. Burks. Computation, behaviour, and structure in fixed and growing automata. In *Self–organising systems*, 1960.

[28] R. Burstall. Program proving as hand simulation with a little induction. In *Proc. of IFIP Congress 74.* ,North–Holland, 1974.

[29] N. Busi, R. Gorrieri, and G. Siliprandi. Distributed conflicts in communicating systems. Technical report, University of Bologna technical report, 1994.

[30] D. Bustard, J. Elder, and J. Welsh. *Concurrent Program Structures.* Prentice–Hall, 1988. ISBN 0–13–167289–4.

[31] K. Forward C. Ho, K. Parker. Petri net modelling for hardware implementation of communication protocols. In *Proc. 9th Australian Microelectronics Conf.*, 1990.

[32] K. Calvert and S. Lam. Adaptors for protocol conversion. In *INFOCOM '90.* IEEE Computer Society Press, 1990.

[33] K. Calvert and S. Lam. Formal methods for protocol conversion. *IEEE Selected Areas in Comms.*, 8(1), 1990.

[34] R. Castanet, A. Dupeux, and P. Guitton. Ada : A well suited language for specification and implementation of protocols. In *Protocol Specification Testing and Verification V.* ,North–Holland, 1986. Proc. IFIP WG 6.1.

[35] A. Cavalli and F. Horn. Proof of specification properties by using finite state machines and temporal logic. In *Protocol Specification Testing and Verification VII.* ,North–Holland, 1987. Proc. IFIP WG 6.1.

[36] A. Cavalli, S. Kim, and P. Maigron. Improving conformance testing for LOTOS. In *FORTE '93*, 1993.

[37] CCITT. Functional specification and description language – SDL. CCITT Recommendations Z100 – Z104, 1984.

[38] CCITT. IPS–OSI specification of ASN.1 – recommendation X.208. Technical report, CCITT, Blue Book 1988.

[39] K. Chandy and J. Misra. *Parallel Program Design : A Foundation.* Addison–Wesley, 1988. ISBN 0–201–05866–9.

[40] J. Chang and M. Liu. Using protocol validation techniques to solve protocol conversion problems. In *Proc. 9th Annual Int. Conf. on Computers and Communications.* IEEE, 1990.

[41] Z. C. Chen and C. A. R . Hoare. Partial correctness of communicating processes and protocols. Research monograph PRG 20, Oxford University Programming Research Group, 1981.

[42] C. H. Chow. Protocol issues in interconnecting ISDN, IN and LAN. In *TENCON '90.* IEEE, 1990.

[43] T. Chow. Testing software design modelled by finite state machines. *IEEE Trans Software Eng.*, 4, 1978.

[44] E. Clarke et al. Using temporal logic for automatic verification of finite state systems. In *Logics and Models for Concurrent Systems.* NATO ASI Series, 1985.

[45] E. Codd. *Cellular Automata.* Academic Press, 1968.

[46] E. Coffman and G. Lueker. *Probalistic Analysis of Packing and Partitioning Algorithms.* Discrete Maths and Optimisation. Wiley, 1991. ISBN 0–471–53272–X.

[47] B. Cohen. Justification of formal methods for system specification. *Software and Microsystems*, 1(5), 1982.

[48] M. Cookson. Application of LOTOS conformance tests : Msc thesis. Master's thesis, BT Laboratories, 1991.

[49] D. Craigen. Tool support for formal methods. In *Proc. 13th Int. Conf. of Software Engineering*, 1991.

[50] A. Dahbura, K. Sabnani, and M. Uyar. Formal methods for generating protocol conformance test sequences. *Proc. of the IEEE*, 78(8), 1990.

[51] V. Dahl and P. Saint-Dizier. *Natural Language Understanding and Logic Programming.* North–Holland, 1984.

[52] D. Davies, C. Hilsum, and A. Rudge, editors. *Communications After AD2000*, pages 215–224. Chapman and Hall, 1993.

[53] T. Denvir. *Software Engineering Mathematics.* Macmillan, 1986.

[54] N. Deo. *Graph Theory with Applications.* Prentice–Hall, 1974.

[55] H. Dixon. Superhighways sans frontiers. *Financial Times*, 21st Feb., 1994.

[56] R. Dssouli and G. Bochmann. Conformance testing with multiple observers. In *Protocol Specification Testing and Verification VI.* ,North–Holland, 1987. Proc. IFIP WG 6.1.

[57] D. Ince (Ed). *Mechanical Intelligence : The Collected Papers of Alan Turing.* Prentice–Hall, 1994.

[58] Lapedes (Ed). *Dictionary of Mathematics and Physics.* McGraw–Hill, 1978.

[59] S. Shapiro (Ed). *Encyclopedia of Artificial Intelligence.* Wiley, 1987.

[60] H. Ehrig, W. Fey, and H. Hansen. ACT ONE : An algebraic specification language with two levels of semantics. Technical report, Nr. 83–03 Institut fur Software und Theoretische Informatik Technische Universitat Berlin, 1983.

[61] H. Ehrig and others. Algebraic concepts for software development in ACT ONE, ACT TWO and LOTOS. In *Proceedings Software–Entwicklung.* Springer–Verlag, 1989.

[62] P. Eijk, H. Kremer, and M. Sinderen. On the use of specification styles for automated protocol implementation from LOTOS to C. In *Protocol Specification Testing and Verification X.* ,North–Holland, 1990. Proc. IFIP WG 6.1.

[63] D. Kristol et al. A polynomial algorithm for gateway generation from formal specifications. *IEEE Trans on Networking*, 1:217–229, 1993.

[64] S. Li et al. An ILAN–ISDN gateway. In *ICC '88*. IEEE, 1988.

[65] T. Cheung et al. Generating test sequences and their degrees of indeterminism for protocols. In *PSTV XI*, 1991.

[66] Lin F. and M. Liu. The rise of protocol engineering. *IEEE Software*, 1992.

[67] O. Faergemand and M. Marques (eds). *SDL'89 – The Language at Work.* ,North–Holland, 1989. ISBN 0–444–88337–1.

[68] J. Favreau and R. Linn. Automatic generation of test scenario skeletons from protocol specifications written in ESTELLE. In *Protocol Specification Testing and Verification VI.* ,North–Holland, 1987. Proc. IFIP WG 6.1.

[69] J. Feigenbaum. Directed Cartesian–product graphs have unique factorisations that can be computed in polynomial time. *Discrete Applied maths*, 15:105–110, 1986.

[70] J. Feigenbaum, J. Hershberger, and A. Schaffer. A polynomial time algorithm for finding the prime factors of cartesian–product graphs. *Discrete Applied maths*, pages 123–138, 1985.

[71] A. Fekete. Formal methods of communication services : A case study. *Computer*, pages 37–47, August 1993.

[72] P. Fencott, M. Lockyer, and P. Taylor. Experiences in integrating structured and formal notations for real–time systems. Technical report, Teeside Polytechnic, 1991.

[73] E. Fergus. *Specifying Distributed Applications : The Limits of Formality.* PhD thesis, Open University, 1989.

[74] J. Findlay. Time : A treatment of some puzzles. *Australian Journal of Philosophy*, 19, 1941.

[75] A. Fleischmann. PASS – A technique for specifying communication protocols. In *Protocol Specification Testing and Verification VII.* ,North–Holland, 1987. Proc. IFIP WG 6.1.

[76] R. Floyd. Assigning meaning to programs. In *Proc. of Symp. in Applied Maths.* American Mathematical Society, 1967.

[77] G. Fowler, R. Haralick, and F. Gray. Feustel C. Grinstead C. Efficient graph automorphism by vertex partitioning. *Artificial Intelligence*, 21(1), 1983.

[78] N. Francez. *Fairness.* Springer–Verlag, 1986. ISBN 0–387–96235–2.

[79] D. Freestone. Specification and verification in communication standards. Technical report, BT Laboratories RT62/006/88, 1988.

[80] A. Galton. Temporal logic and computer science : An overview. In *Temporal Logics and their Applications.* Academic Press, 1987.

[81] J. Goguen and T. Winkler. *Introducing OBJ3.* SRI International, 1988.

[82] W. Golson and J.W. Rounds. Connections between two theories of concurrency : Metric spaces and synchronisation trees. *Information and Control*, 57, 1983.

[83] G. Gonenc. A method for the design of fault detection experiments. *IEEE Trans. Comp.*, 19, 1970.

[84] A. Gore. Superhighways – personal view. *Financial Times*, 19th Sept., 1994.

[85] R. Gotzhein. Specifying communication services with temporal logic. In *Protocol Specification Testing and Verification X.* ,North–Holland, 1990. Proc. IFIP WG 6.1.

[86] P. Green. Protocol conversion. *IEEE Trans on Comm.*, 34(3), 1986.

[87] EC Information Society Group. Europe and the gobal information society. Technical report, EC, 1994.

[88] J. Guttag. Abstract data types and the development of data structures. *ACM*, 20(6), 1977.

[89] O. Haas. Formal protocol specification based on attribute grammars. In *Protocol Specification Testing and Verification V.* ,North–Holland, 1986. Proc. IFIP WG 6.1.

[90] A. Hall. Seven myths of formal methods. *IEE Software*, 1990.

[91] V. Hamilton. Experience of combining Yourdon and VDM. Technical report, Rolls–Royce and Associates LTD., 1991.

[92] J. Harangozo. An approach to describing the link protocol with a formal language. In *Proc. 5th Data Comms. Symp.*, 1977.

[93] F. Harary. *Graph Theory.* Addison–Wesley, 1969.

[94] D. Harel. *Algorithmics.* Addison–Wesley, 1987.

[95] M. Hennessy. *Algebraic Theory of Processes.* MIT Press, 1988. ISBN 0–262–08171–7.

[96] L. Herault and Jean-Jacques Niez. How neural networks can solve hard graph problems. In *Proc. Neural Networks 89*, 1989.

[97] C. Hoare and J. Shepherdson. *Mathematical Logic and Programming Languages.* Prentice–Hall, 1985.

[98] C.A.R. Hoare. An axiomatic basis for computer programming. *Comms. of the ACM*, 12, 1969.

[99] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice–Hall, 1985.

[100] C.A.R. Hoare. An overview of some formal methods for program design. *IEEE Computer*, 1987.

[101] M. Hoffman. Hardware implementation of communication protocols : A formal approach. In *Proc. 7th Annual Symp. on Computer Arch.* Hoffman, 1980.

[102] G. Holzmann. On limits and possibilities of automated protocol analysis. In *Protocol Specification Testing and Verification VII.* ,North–Holland, 1987. Proc. IFIP WG 6.1.

[103] G. Holzmann. Protocol design : Redefining the state of the art. *IEEE Software*, 1992.

[104] J. Hopcroft and J. Ullman. *Introduction to Automata Theory Languages and Computation.* Addison–Wesley, 1979. ISBN 0–201–02988–X.

[105] C. Huang and M. Liu. An incremental protocol test method: Formal modelling and architectures. In *ICSI '92.* IEEE Computer Society Press, 1992.

[106] H. Igarashi, Y. Kakuda, and T. Kikuno. Synthesis of protocol specifications for design of responsive protocols. *IEICE Trans. on Information and Systems*, pages 1375–1385, 1993.

[107] W. Imrich and J. Zerovnik. Factoring Cartesian product graphs. *Journal of Graph Theory*, 18(6), 1994.

[108] D. Ince. *Software Development : Fashioning the Baroque.* OUP, 1988.

[109] The Independent. Customers urged to check credit bills, 1994.

[110] ISO. LOTOS. Technical report, International Standards Organisation, Geneva, 1984.

[111] ISO. ISO TC97/SC21/WG16–1 estelle – a formal description technique based on an extended state transition model. Technical report, ISO, 1985.

[112] ISO. ISO/IEC DIS 9646 Information Technology – Open Systems Interconnection – conformance testing methodology and framework. Technical report, ISO, 1990.

[113] M. Jackson. Power and limitations of formal methods. *Journal of Information Technology*, 2(2), 1987.

[114] J.F.DeLapeyre. Introduction to protocol engineering using specification and description language. *BT Technology Journal*, 11(4):7–8, 1993.

[115] J.F.Kurose and Y.Yemini. Specification and verification of a connection establishment protocol using temporal logic. In *Protocol Specification, Testing and Verification.* ,North–Holland, 1982.

[116] C.B. Jones. *Systematic Software Development Using VDM.* Prentice–Hall, 1986.

[117] Y. Kakuda, H. Igarashi, and T. Kikuno. Automated synthesis of protocol specifications with message collisions and verification of timeliness. Technical report, Department of Computer Science, Osaka University, 1–3 Machikaneyama–cho, Toyonaka–shi, Osaka 560, Japan, 1994.

[118] Y. Kakuda, M. Nakamura, and T. Kikuno. Automated synthesis of protocol specifications from service specifications with parallelly executable multiple primitives. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Science*, pages 123–138, 1994.

[119] A. Kay and J. Reed. Using rely and guarantee in timed CSP. Technical report, Oxford Univ. PRG, 1991.

[120] S. Knapskog. Formal specification and verification of secure communication protocols. In *Advances in Cryptology - AUSCRYPT '90.* Springer–Verlag, 1990.

[121] P. Kritzinger. Protocol performance using image protocols. In *Protocol Specification Testing and Verification VII.* ,North–Holland, 1987. Proc. IFIP WG 6.1.

[122] R. Lai. Teaching communication protocol development using formal methods. *ACCIS*, 1(2), 1994.

[123] J. Lam and J. Delosme. Simulated annealing : a fast heuristic for some generic layout problems. In *IEEE Int. Conf. on CAD.* IEEE, 1988.

[124] S. Lam. Protocol conversion : Correctness problems. In *Proc. ACM SIGCOMM'86 Symp.* ACM, 1986.

[125] S. Lam. Protocol conversion. *IEEE Trans. Software Eng.*, 14(3), 1988.

[126] S. Lam and A. Shanker. Protocol verification via projections. *IEEE Trans on Software Eng.*, 10(4), 1984.

[127] S. Lam and A. Shanker. A theory of interfaces and modules i – composition theorem. *IEEE Trans on Software Eng.*, 20(1), 1994.

[128] L. Lamport. While waiting for the millenium : Formal specification and verification of concurrent systems now. In *Concurrency 88.* Springer–Verlag, 1988.

[129] D. Lehmann, A. Pnueli, and J. Stavi. Impartiality justice and fairness : The ethics of concurrent termination. In *Proc. 8th Coll. on Automata Languages and Programming.* Springer–Verlag, 1981. 115.

[130] L. Li. A formal specification technique for communication protocol. In *INFOCOM'89*. IEEE, 1989.

[131] M. Li and N. Georganas. Coloured generalised stochastic Petri nets for integrated systems protocol performance modelling. *Computer Communications*, 13(7), 1990.

[132] X. Li, R. Lai, and T. Dillon. Theory of deductive systems for protocol verification. In *ICCI '92*. IEEE Computer Society Press, 1992.

[133] F. Lin, P. Chu, and M. Liu. Protocol verification using reachability analysis. *ACM SIGCOMM Computer Comm. Rev.*, 15(5), 1987.

[134] F. Lin and M. Liu. Protocol validation for large scale applications. *IEEE Software*, 1992.

[135] F. J. Lin and M. T. Liu. Protocol validation for large–scale applications. *IEEE Software*, 9(1):23–26, 1992.

[136] R. Linn. The features and facilities of ESTELLE. In *Protocol Specification Testing and Verification V.* ,North–Holland, 1986. Proc. IFIP WG 6.1.

[137] A. Macro and J. Buxton. *The Craft of Software Engineering.* Addison–Wesley, 1987.

[138] G. Martin. An algorithm for determining observational equivalence of CCS agents. Technical report, British Telecom Research and Technology Internal Memorandum Number R11/87/016, 1987.

[139] G. Martin. An algorithm for determining observational equivalence of CCS agents. Technical report, British Telecom Research and Technology Internal Memorandum Number RT31/87/016, 1987.

[140] G. Martin. A general solution to the interface equation. Technical report, British Telecom Research and Technology Internal Memorandum Number R11/87/023, 1987.

[141] G. Martin. Practical considerations of applying the interface equation. Technical report, British Telecom Research and Technology Internal Memorandum Number R11/87/007, 1987.

[142] G. Martin. A study into a constructive method for generating solutions to the interface equation. Technical report, British Telecom Research and Technology Internal Memorandum Number RT31/89/004, 1989.

[143] G. Martin. A study into a constructive method for generating solutions to the interface equation. Technical report, British Telecom Research and Technology Internal Memorandum Number RT31/89/004, 1989.

[144] G. Martin and A. Pengelly. A note describing the relevance of the merge/split routine to the constructive algorithm for solving the interface equation and ideas for its implementation. Technical report, BT Labs, Martlesham Heath, Ipswich IP5 7RE, UK., 1993.

[145] M.D.Cookson and S.G.Woodsford. Design methodology using SDL. *BT Technology Journal*, 11(4):16–24, 1993.

[146] J. Meer. Derivation and validation of test scenarios based on the formal specification language LOTOS. In *Protocol Specification Testing and Verification VI.* ,North–Holland, 1987. Proc. IFIP WG 6.1.

[147] K. Meinke and J. Tucker. The scope and limits of synchronous concurrent computation. In *Concurrency 88.* Springer–Verlag, 1988.

[148] J. Meseguer. A logical theory of concurrent objects. In *Proc. Object Oriented Programming.* ACM, 1990.

[149] D. Meyer. The graph topology on nth–order systems is quotient euclidean. *IEEE Trans Automatic Control*, 36(3), 1991.

[150] R. Miller. Protocol verification : The first ten years the next ten years. In *Protocol Specification Testing and Verification X.* ,North–Holland, 1990. Proc. IFIP WG 6.1.

[151] R. Milner. Synthesis of communicating behaviour. *LNCS*, 64, 1978.

[152] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science.* Springer–Verlag, 1980. ISBN 3–540–10235–3.

[153] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[154] R. Milner. *Communication and Concurrency.* Computer Science. Prentice–Hall, 1989. ISBN 0–13–114984–9.

[155] F. Moller. A temporal calculus of communicating systems. In *Proceedings of CONCUR'90 : LNCS 458.* Springer–Verlag, 1990.

[156] F. Moller. The Edinburgh Concurrency Workbench (version 6.1). Technical report, Department of Computer Science, University of Edinburgh, 1992.

[157] Maxemchuk N and K. Sabnani. Probabilistic verification of communication protocols. In *Protocol Specification Testing and Verification VII.* ,North–Holland, 1987. Proc. IFIP WG 6.1.

[158] S. Naeher. *The LEDA graph analysis tool.* Max–Planck–Institut fuer Informatik, Im Stadtwald, 6600 Saarbruecken, FRG.

[159] K. Naik and B. Sarikaya. Testing communication protocols. *IEEE Software*, 9(1):27–37, 1992.

[160] S. Naito and M. Tsunoyama. Fault detection for sequential machines by transition tours. In *Proc. IEEE Fault Tolerant Comp. Conf.* IEEE, 1981.

[161] E. Najm. A verification oriented specification in LOTOS of the transport protocol. In *Protocol Specification Testing and Verification VII.* ,North–Holland, 1987. Proc. IFIP WG 6.1.

[162] T. Nakakawaji et al. Application of Concurrent Euclid to specification, implementation and verification of communication protocols. In *Protocol Specification Testing and Verification VI.* ,North–Holland, 1987. Proc. IFIP WG 6.1.

[163] P.A. Ng and R.T. Yeh. *Modern Software Engineering.* Van–Nostrand, 1989.

[164] M.T. Norris. The role of formal methods in system design. *BT Tech J*, 3(4), 1985.

[165] M.T. Norris, R. Everett, G. Martin, and M. Shields. A method for the synthesis of interactive system specifications. In *Proc Global Telecom. Conference Miami*, 1988.

[166] M.T. Norris, M.W. Shields, and J. Ganeri. A theoretical basis for the construction of interactive systems. *BT Tech J*, 5(2), 1987.

[167] Y. Ohara, S. Yoshitake, and T. Kawaoka. Protocol conversion method for heterogeneous systems interconnection in multi–profile environment. In *Protocol Specification Testing and Verification VII.* ,North–Holland, 1987. Proc. IFIP WG 6.1.

[168] K. Okumara. Generation of proper adapters and converters from a formal service specification. In *INFOCOM '90.* IEEE Computer Society Press, 1990.

[169] K. Okumura. A formal protocol conversion method. In *Proc. ACM SIGCOMM'86 Symp.*, 1986.

[170] J. Oxtoby and B. Pettis. John von neumann 1903–1957. *Bulletin of the American Mathematical Society*, 64(3), 1958.

[171] J. Pachl. Protocol description and analysis based on a state transition model with channel expressions. In *Protocol Specification Testing and Verification VII.* ,North–Holland, 1987. Proc. IFIP WG 6.1.

[172] K. Paliwoda and J. Sanders. A sliding window protocol in CSP. Research monograph PRG 66, Oxford University Programming Research Group, 1988.

[173] D. Park. Concurrency and automata on infinite series. In *Proc 5th GI Conference.* Springer, 1981. Volume 104 of Lecture Notes in Computer Science.

[174] K. Parker, R. Berger, and M. Malgeri. Protocol analysis and implementation using NPNs and SDL. In *Specification and Verification of Concurrent Systems.* Spriner–Verlag, 1990.

[175] J. Parrow. Submodule construction as equation solving in CCS. Technical report, Laboratory for the Foundations of Computer Science, University of Edinburgh, 1987.

[176] B. Pehrson. Protocol verification for OSI. *Computer Networks and ISDN Systems*, 18(3), 1990.

[177] A. Pengelly. An information theoretic measure of system structure. Technical report, BT Labs, Martlesham Heath, UK, Technical Report., 1989.

[178] A. Pengelly and D. Ince. Quotients machines and the interface equation. Technical report, Open University, Maths Faculty, 1994.

[179] J. Peterson. *Petri Net Theory and the Modelling of Systems.* Prentice–Hall, 1981.

[180] M. Peyravian and C. Lea. An algorithmic method for protocol conversion. In *Proc. 12th Int. Conf. on Distributed Comp. Systems*, 1992.

[181] T. Piatkowski, L. Ip, and D. He. State Architecture Notation and simulation : A formal technique for specification and testing of protocol systems. *Computer Networks*, 6, 1982.

[182] D. Pitt and D. Freestone. The derivation of conformance tests from LOTOS specifications. *IEEE Trans. Software Eng.*, 16(12), 1990.

[183] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13, 1981.

[184] Fault Tolerance : Principles and Practice. *T. Anderson and P. Lee.* Prentice–Hall, 1981.

[185] A. Prior. Diodorian modalities. *Philosophical Quarterly*, 5, 1955.

[186] W. Quine. *Elementary Logic.* Harper and Row, 1965.

[187] M. Rajagopal and R. Miller. Synthesizing a protocol converter from executable protocol traces. *IEEE Trans. on Computers*, 40(4), 1991.

[188] L. Ranatunga. Process synthesis and the context equation of depth one. Technical report, Electronic Engineering Laboratories University of Kent, 1989.

[189] L. Ranatunga. System partitioning by solving an equation in CCS. Technical report, Electronic Engineering Laboratories University of Kent, 1989.

[190] V.J. Rayward-Smith. *A First Course in Abstract Language Theory.* Blackwell, 1984.

[191] J. Reggia, S. Armentrout, H.H. Chou, and Y. Peng. Simple systems that exhibit self-directed replication. *SCIENCE*, 259:1282–1287, 1993.

[192] G. Reinhard and G. Bochmann. Deriving protocol specifications from service specifications including parameters. *ACM Tans. Computer Systems*, 8(4), 1990.

[193] W. Reisig. Temporal logic and causality in concurrent systems. In *Concurrency 88*. Springer–Verlag, 1988.

[194] J. Risbridger and D. Stewart. A protocol tester for message based signalling systems. *BT Technology Journal*, 5(2), 1987.

[195] D. Robinson. *Digraphs*. Gordon and Breach, 1980.

[196] A. Rockstrom. *An Introduction to CCITT SDL*. CCITT, 1985. ISBN 91–7810–321–5.

[197] C. Roisin. Protocol description with the Occam language. In *Protocol Specification Testing and Verification V.* ,North–Holland, 1986. Proc. IFIP WG 6.1.

[198] W. Rounds. Applications of topology to semantics of communicating processes. In *Proc. NSF–SERC Int Symp. on Concurrency*. Springer–Verlag, 1985. LNCS 197.

[199] W. Rounds. On the relationship between scott domains synchronisation trees and metric spaces. *Information and Control*, 6(28), 1985.

[200] G. Rucker and C. Rucker. On using the adjacency matrix power method for perception of symmetry and for isomorphism testing of highly intricate graphs. *Journal of Chemical Information and Computer Science*, 31(1), 1991.

[201] H. Rudin. An informal overview of formal protocol specification. *IEEE Communications Magazine*, 23(3):46–52, 1985.

[202] K. Sabnani and A. Dahbura. A protocol test generation procedure. *Comp. Networks and ISDN Sys.*, 15, 1988.

[203] D. Saha and P. Dhar. A fast protocol conversion technique using reduction of state transition graphs. In *11th Annual Int. Phoenix Conf. on Computers and Communication*. IEEE, 1992.

[204] M. Sajkowski. Protocol verification in the presence of time. In *Protocol Specification Testing and Verification VI.* ,North–Holland, 1987. Proc. IFIP WG 6.1.

[205] K. Saleh. Automatic synthesis of protocol specifications from service specifications. In *Proc. Int'l. Phoenix Conference on Computers and Communications*, 1991.

[206] K. Saleh and R. Probert. Synthesis of communication protocols: Survey and assessment. *IEEE Trans. on Computers*, 40(4):468–475, 1991.

[207] R. Saracco, J. Smith, and R. Reed. *Telecommunications Systems Engineering and SDL.* ,North–Holland, 1989. ISBN 0–444–88084–4.

[208] R. Sarnath and H. Xin. A p–complete graph partitioning problem. *Theoretical Computer Science*, 76(2), 1990.

[209] J. Savage and M. Wloka. Parallelism in graph-partitioning. *Journal of Parallel and Distributed Computing*, 13:257–272, 1991.

[210] R. Scharwtz and P. Melliar-Smith. From state machines to temporal logic : Specification methods for protocol standards. *IEEE Trans. on Comm.*, 30, 1982.

[211] S. Schindler. The OSA Project : Basic concepts of formal specification techniques and rspl. Technical report, Tech. Univ. Berlin Tech. Report 80–13, 1980.

[212] C.E. Shannon. Mathematical theory of communication. *BSTJ*, 27(379), 1948.

[213] D. Shasha, A. Pneulli, and W. Ewald. Temporal verification of carrier–sense local area network protocols. In *Proceedings of the 11th ACM Symposium on the principles of programming languages*, 1983.

[214] D. Shepherd. Formal design techniques in the implementation of concurrency. In *Colloquium on Formal Methods for Protocols.* IEE, 1991. Digest No. 1991/043.

[215] M. Shields. Concurrent machines. *Computer Journal*, 28(5), 1985.

[216] M. Shields. *An Introduction to Automata Theory.* Blackwell, 1987. ISBN 0–632–01756–2.

[217] M . W. Shields. Implicit system specification and the interface equation. *The Computer Journal*, 32(5), 1989.

[218] M. W. Shields. Solving the interface equation Tech. Rep. SE/079/2. Technical report, Electronic Engineering Laboratories University of Kent at Canterbury, July 1986.

[219] M.W. Shields. Extending the interface equation Tech. Rep. SE/079/3. Technical report, Electronic Engineering Laboratories University of Kent at Canterbury, August 1986.

[220] J. Shu and M. Liu. A synchronisation model for protocol conversion. In *Proc. ACM SIGCOMM'86 Symp.*, 1986.

[221] J. Shu and M. Liu. Protocol conversion between complex protocols. In *Proc. 9th Annual Int. Conf. on Computers and Communications.* IEEE, 1990.

[222] D. Sidhu. Protocol testing : The first ten years the next ten years. In *Protocol Specification Testing and Verification X.* ,North–Holland, 1990. Proc. IFIP WG 6.1.

[223] D. Sidhu and T. Leung. Formal methods for protocol testing : A detailed study. *IEEE Trans. Software Eng.*, 15(4), 1989.

[224] J. Small. *The GAP graph analysis tool.* jsmall@marlin.nosc.mil (Joel F. Small).

[225] Engineering Quality Software. *D. Smith and K. Wood.* Elsevier, 1987.

[226] M. J. Spivey. *Understanding Z.* Cambridge, 1988. ISBN 0–521–33429–2.

[227] K. Sridhar and R. Bharadwaj. CSP as a formal description technique for OSI protocols. In *Ibericom'87.* Assoc. Portuguesa Inf., 1987.

[228] F. Stomp. A derivation of a broadcasting protocol using sequentially phased reasoning. In *Protocol Specification Testing and Verification X.* ,North–Holland, 1990. Proc. IFIP WG 6.1.

[229] F. Stomp and W. de Roever. Designing distributed systems by means of formal sequentially phased reasoning. In *Proc. 3rd Int. Conf. on Distributed Algorithms.* Springer Verlag, 1989. LNCS 392.

[230] X. Sun, Y. N. Shen, and F. Lombardi. On the verification and validation of protocols with high fault coverage using UIO sequences. In *Proceedings 11th Symposium on Reliable Distributed Systems.* IEEE Computer Society Press, 1992.

[231] P. Tilanus and Y. Yang. Experience with LOTOS and environment LOTTE on an ISDN protocol. In *Specification and Verification of Concurrent Systems.* Spriner–Verlag, 1990.

[232] D. Till and B. Potter. The specification in Z of gateway functions within a communications network. In *Proc. IFIP Distributed Processing.* ,North–Holland, 1988.

[233] M. J. Toussaint. A new method for analysing the security of cryptographic protocols. *IEEE Journal on Selected Areas in Communications*, 11(5), 1993.

[234] K. Turner. The use of formal methods in communication standards. In *Colloquium on Formal Methods for Protocols.* IEE, 1991. Digest No. 1991/043.

[235] S. Ueno, K. Tsuji, and Y. Kajitani. A note on dual trail partition of a plane graph. *IEICE Transactions*, 74(7), 1991.

[236] R. Venkatraman and T. Piatowski. A formal comparison of formal protocol specification techniques. In *Protocol Specification Testing and Verification V.* ,North–Holland, 1986. Proc. IFIP WG 6.1.

[237] G. Vijayan. Partitioning logic to optimize routability on graph structures. In *1990 IEEE Int. Symp. on Circuits and Systems.* IEEE, 1990.

[238] C. Vissers and L. Logrippo. The importance of the service concept in the design of data communications protocols. In *Proc. IFIP WG 6.1.* ,North–Holland, 1986.

[239] C. Walker. Is it safe to fly by wire? *Computer Weekly*, 1995.

[240] J. Walker. Strict refinement of graphs and digraphs. *J. Combinatorial Theory*, 43, 1987.

[241] C. J. Wang and M. T. Liu. Axiomatic test sequence generation for EFSMs. In *Proceedings 12th Int. Conf. on Distributed Computing Systems.* IEEE Computer Society Press, 1992.

[242] M. Warner. Pseudomorphisms of automata. *Kybernetika*, 19(1):325–339, 1988.

[243] M. Werner. *The daVinci graph analysis tool.* Institute for Formal Methods in Software Engineering, University of Bremen, Postfach 330440,D–28334 Bremen.

[244] C. West. Protocol validation by random state exploration. In *Protocol Specification Testing and Verification VI.* ,North–Holland, 1987. Proc. IFIP WG 6.1.

[245] C. West. The first ten years – the next ten years. In *Protocol Specification Testing and Verification X.* ,North–Holland, 1990. Proc. IFIP WG 6.1.

[246] C. Wezeman. The CO–OP Method A Method for Compositional Derivation of Canonical Testers. MSc Thesis. Master's thesis, University of Twente, 1989.

[247] C. Wezeman. The full LOTOS CO–OP method. Technical report, LOTOSPHERE – ESPRIT 2304, 1990.

[248] M. Wheeler. Numerical Petri nets – technical report 7780. Technical report, Telecom Australia Research Labs., 1985.

[249] R. Wilson. *Graph Theory.* Oliver and Boyd, 1972.

[250] P. Winkler. Factoring a graph in polynomial time. *Europ. J. Combinatorics*, 8:209–212, 1987.

[251] P. Winkler. The metric structure of graphs. Technical report, Emory University, Atlanta, Georgia, USA, 1987.

[252] H. Xin, Z. Hong, and C. Xiyao. Heuristic software partitioning algorithms for distributed real–time applications. In *Proceedings 10th Int. Conf. Software Eng.* IEEE, 1988.

[253] Y.W. Yao, W.S. Chen, and M.T. Liu. A modular approach to constructing protocol converters. In *Proceedings IEEE INFOCOM'90*, 1990.

[254] Y.W. Yao, W.S. Chen, and M.T. Liu. A parallel model for the construction of protocol converters. In *Proceedings GLOBECOM'90 volume 3*, 1990.

[255] V. Yodaiken and K. Ramamritham. *Verification of a reliable net protocol.* Springer–Verlag, 1992.

[256] J. Young. *Information Theory.* Butterworth, 1971.

A Notation

The format of this glossary is *subject - description - example.*

'$\alpha, \beta \ldots$' –CCS actions.

'$\overline{\alpha}, \overline{\beta} \ldots$' – CCS complementary actions.

'$\overline{\mu}$' Complementary or dual actions synchronise with their uncomplemented versions when two CCS agents are composed in parallel.

'τ' The invisible action, $p = \alpha.q + \tau.r$.

'τ_i **and** τ_c' – Intrinsic and communicating invisible actions, see Def 5.3.7.

'$p, q, p', q' \ldots$' – CCS agents, $p = \alpha.q + \beta.r$.

'*NIL*' The agent which cannot perform any action, $p = \alpha.NIL$ *i.e.* p performs an α action and then stops.

's' Any sequence of actions, $p \Rightarrow^s q$.

'ϵ' The empty sequence of actions, $p \Rightarrow^\epsilon p$.

'K, $L \ldots$' Various sets, used to denote sets in $p - q$ and $p - r$ space respectively.

'Π, $\tilde{\Pi}$**-planes** ' – Partition of $\mathcal{M}_q$ into disjoint planes, see Def 5.3.12

'$\emptyset$' The empty set.

'+' CCS Choice operator, $p = \alpha.q + \beta.r$: p may perform an α next or it may perform a β.

'\' CCS hiding operator, $p \backslash A$ is an agent derived from p by rendering invisible the actions (and their complements) defined by the set of actions A.

'$\wr$' – CCSg equivalent to \, Removes communicating actions on formation of τ_c's. See Def 6.2.10.

'$\times$' Graph Cartesian Product.

'$\otimes$' – Group direct product.

'**Sym**' Symmetric closure, $\mathbf{Sym}(\mathcal{M}_p)$.

'Υ' – Tau splitting, see Def 5.3.14.

'σ' – Mapping from $\mathcal{M}_q$ to $\mathcal{M}_p$.

'|' CCS parallel composition, $(A|B)$ is the agent made by composing A and B concurrently, according to the conventions of CCS.

'||' – CCSg equivalent to |, only actions specified by $\wr$ will communicate. See Def 6.2.8.

'$\rightarrow^\mu$' has an immediate μ derivation to, $p \rightarrow^\mu q$ p may next perform a μ action and then become q. Note: the μ is a single action, possibly the invisible τ action.

'$\Rightarrow^s$' has a visible s-sequence derivation, $p \Rightarrow^s q$ Note: the s denotes a possibly empty sequence of *visible* actions (*i.e.* excluding τ). The statement $p \Rightarrow^s q$ means that if s is the sequence $\alpha, \beta \ldots$, then there exists at least one sequence of derivations like $p \rightarrow^\tau p_1 \ldots \rightarrow^\tau p_2 \rightarrow^\alpha p_3 \rightarrow^\tau \ldots \rightarrow^\beta p_4 \ldots q$, in which there may be any number (possibly zero) of τ-derivations interspersed between the p, the $\alpha, \beta \ldots\ldots$ and the q.

'$\approx$' is observationally equivalent to, $p \approx q$.

'$\sim$' – is strongly equivalent to, $p \sim q$.

'$\stackrel{S}{=}$' – is structurally equivalent to, $p \stackrel{S}{=} q$. See Def 6.2.5.

'$\cong$' – graph isomorphism, $\mathcal{G}_{\mathcal{M}_p} \cong \mathcal{G}_{\mathcal{M}_r}$.

'$\Lambda(\ldots)$' The set of actions of ..., $\Lambda(p)$.

'trace' The set of all possible sequences $\mu_1 \ldots \mu_n \in \Lambda(\ldots)$ $n \geq 0$, that is, every conceivable sequence of actions (including none at all) that can be made by stringing together actions of the agent.

'A' – Set containing actions which can communicate, $\backslash A$ or $\wr A$.

'B' – Set containing actions which cannot communicate, $\backslash A$.

'C' – Equal to $A \cup B$.

'R^{∞}' – Set of states reachable via infinite length paths.

'Δ' – Machine difference in terms of state and action sets, $\Delta(\mathcal{M}_p, \mathcal{M}_r)$.

'$\mathcal{M}_x$' – Generic representation of a machine.

'$R(\ldots)$' – The set of all states reachable from ..., $R(p)$.

S – A system (set) of K-sets which is a potential solution.

$\Re$ – Elementary sets from which K-sets are derived.

$N_{\mu,K}$, $N_{\tau,K}$ – Sets which act as devices in generating $\Re$ sets.

ψ – The set of all K-sets in $P(R(p) \times R(q))$, $K \subset \psi(p,q)$.

$\sim_S^*$ – An equivalence relation over the set S, used in the constructive algorithm.

B Example from Chapter 5

The $\mathcal{M}_p$ and $\mathcal{M}_q$ machines are specified below using CCS prefix notation.

$$
\begin{aligned}
p_1 &\Leftarrow \alpha p_2 + \beta p_3 + \gamma p_4 + \tau p_5 \\
p_2 &\Leftarrow \alpha p_3 + \overline{send2} p_1 + \overline{send1} p_4 \\
p_3 &\Leftarrow \beta p_5 + \gamma p_2 + \tau p_2 + \beta p_2 \\
p_4 &\Leftarrow \gamma p_2 + \overline{send2} p_5 + \overline{send1} p_3 + \overline{send1} p_1 \\
p_5 &\Leftarrow \gamma p_1 + \overline{send2} p_1
\end{aligned}
$$

$$
\begin{aligned}
q_1 &\Leftarrow \alpha q_2 + \beta q_3 + \gamma q_4 + \tau q_5 + \mu q_6 + \nu q_7 + \nu q_8 \\
q_2 &\Leftarrow \alpha q_3 + \mu q_9 + \nu q_{10} + \nu q_{11} \\
q_3 &\Leftarrow \beta q_5 + \gamma q_2 + \tau q_2 + \beta q_2 + \mu q_{12} + \nu q_{13} + \nu q_{14} \\
q_4 &\Leftarrow \gamma q_2 + \mu q_{15} + \nu q_{16} + \nu q_{17} \\
q_5 &\Leftarrow \gamma q_1 + \mu q_{18} + \nu q_{19} + \nu q_{20} \\
q_6 &\Leftarrow \alpha q_9 + \beta q_{12} + \gamma q_{15} + \tau q_{18} \\
q_7 &\Leftarrow \alpha q_{10} + \beta q_{13} + \gamma q_{16} + \tau q_{19} + \nu q_1 \\
q_8 &\Leftarrow \alpha q_{11} + \beta q_{14} + \gamma q_{17} + \tau q_{20} + \mu q_6 + \nu q_1 \\
q_9 &\Leftarrow \alpha q_{12} + \tau q_7 + \tau q_8 \\
q_{10} &\Leftarrow \alpha q_{13} + \nu q_2 + \tau q_{21} \\
q_{11} &\Leftarrow \alpha q_{14} + \mu q_9 + \nu q_2 \\
q_{12} &\Leftarrow \beta q_{18} + \gamma q_9 + \tau q_9 + \beta q_9 \\
q_{13} &\Leftarrow \beta q_{19} + \gamma q_{10} + \tau q_{10} + \beta q_{10} \\
q_{14} &\Leftarrow \beta q_{20} + \gamma q_{11} + \tau q_{11} + \beta q_{11} + \mu q_{12} + \nu q_3 \\
q_{15} &\Leftarrow \gamma q_9 + \tau q_{19} + \tau q_{20} \\
q_{16} &\Leftarrow \gamma q_{10} + \nu q_4 + \tau q_{22} + \tau q_{23} \\
q_{17} &\Leftarrow \gamma q_{11} + \mu q_{15} + \nu q_4 \\
q_{18} &\Leftarrow \gamma q_6 + \tau q_7 + \tau q_8 \\
q_{19} &\Leftarrow \gamma q_7 + \nu q_5 \\
q_{20} &\Leftarrow \gamma q_8 + \mu q_{18} + \nu q_5 \\
q_{21} &\Leftarrow \gamma q_{24} + \nu q_{16} + \tau q_{16} + \tau q_3 + \tau q_1 \\
q_{22} &\Leftarrow \beta q_{25} + \gamma q_{24} + \tau q_{24} + \beta q_{24} + \nu q_{13} + \tau q_{13} \\
q_{23} &\Leftarrow \alpha q_{24} + \beta q_{22} + \gamma q_{21} + \tau q_{25} + \nu q_7 + \tau q_7 \\
q_{24} &\Leftarrow \alpha q_{22} + \nu q_{10} + \tau q_{13} + \tau q_4 \\
q_{25} &\Leftarrow \gamma q_{23} + \nu q_{19} + \tau q_{19}
\end{aligned}
$$

The set of hidden actions A is the set $\{send1, send2\}$.
The $\mathcal{M}_r$ machine satisfying the interface equation $(\mathcal{M}_p \mid \mathcal{M}_r)\backslash A \stackrel{S}{=} \mathcal{M}_q$, is given below.

$$
\begin{aligned}
r_1 &\Leftarrow \mu r_2 + \nu r_3 + \nu r_4 \\
r_2 &\Leftarrow \mathrm{send2} r_3 + \mathrm{send2} r_4 \\
r_3 &\Leftarrow \mu r_2 + \nu r_1 \\
r_4 &\Leftarrow \nu r_1 + \mathrm{send1} r_5 \\
r_5 &\Leftarrow \tau r_4 + \nu r_4 + \mathrm{send1} r_1
\end{aligned}
$$

C Case Study

C.1 Introduction

The following example represents a moderately large conversion/interface problem which possesses a number of characteristics which are meant to extend the theory discussed in the thesis. In terms of the standard examples given in papers etc, it is larger by at least 1.5 orders of magnitude, and possibly two. The example represents the following problem :-

$$
\begin{aligned}
(\mathcal{M}_p^{45} \mid \mathcal{M}_r)\backslash A &\sim \mathcal{M}_q^{540} \\
\text{where } A &\Leftarrow \{\alpha, \delta, \theta, \mu\} \\
\Lambda(\mathcal{M}_p^{45}) &= \{\alpha, \delta, \theta, \iota, \kappa, \mu, \xi, \pi, \sigma, \phi, \psi, \omega\} \\
\Lambda(\mathcal{M}_q^{540}) &= \{\beta, \gamma, \eta, \iota, \kappa, \xi, \pi, \sigma, \tau, \phi, \psi, \omega\} \\
\Lambda(\mathcal{M}_p^{45}) \cap \overline{\Lambda(\mathcal{M}_q^{540})} &\subseteq \tau \\
\Lambda(\mathcal{M}_p^{45}) \cap \overline{A} &= \emptyset \\
\Lambda(\mathcal{M}_q^{540}) \cap (A \cup \overline{A}) &= \emptyset
\end{aligned}
$$

Notice that $\Lambda(\mathcal{M}_q^{540}) \cap (A \cup \overline{A}) \neq \emptyset$

Hence, we expect to derive a machine $\mathcal{M}_r$ with approximately 12 states.

This is a large example, and as such presents a problem in terms of explaining the mechanics of what is happening. As such, only the key issues will be discussed in any great detail.

C.2 Preliminary discussion

The aims of this example were :-

- Examine scalability issues
- Investigate equivalence and contraction problems
- Validate the theory to a high degree
- Uncover, and rectify any problems

$\mathcal{M}_q^{540}$ highlights very well the sort of problems introduced by equivalence notions. With strong equivalence $\mathcal{M}_q^{540}$ is in fact reduced to a machine with 520 states which we will call $\mathcal{M}_q^{520}$. Now since we are dealing with a quotient theory, we expect the order of $\mathcal{M}_p^{45}$, $\mid \mathcal{M}_p^{45} \mid$ to be a divisor of $\mid \mathcal{M}_q^{520} \mid$, which is clearly not the case. Since $\Pi \subset \mathcal{M}_q$ (and there is more than one), we can apply the **Sym** transform to recover the collapsed structure. Since with this example the relative reduction is quite small, the application of symmetric closure worked very well.

We take up the process from the point where the partitioning of $\mathcal{M}_q$ has been completed.

C.3 Walkthrough

$\mathcal{M}_p^{45}$ is defined as :-

$$
\begin{aligned}
p_1 &\Leftarrow \iota p_1 + \kappa p_4 + \xi p_{10} \\
p_2 &\Leftarrow \sigma p_1 + \alpha p_5 \\
p_3 &\Leftarrow \kappa p_2 + \iota p_3 + \xi p_{19} \\
p_4 &\Leftarrow \alpha p_5 + \sigma p_7 \\
p_5 &\Leftarrow \alpha p_4 + \delta p_2 + \alpha p_6 + \mu p_8 \\
p_6 &\Leftarrow \theta p_5 + \sigma p_3 \\
p_7 &\Leftarrow \iota p_7 + \kappa p_8 + \xi p_{28} \\
p_8 &\Leftarrow \alpha p_5 + \sigma p_9 \\
p_9 &\Leftarrow \kappa p_6 + \iota p_9 + \xi p_{37} \\
p_{10} &\Leftarrow \phi p_{13} + \xi p_1 \\
p_{11} &\Leftarrow \sigma p_{10} + \alpha p_{14} \\
p_{12} &\Leftarrow \phi p_{11} \\
p_{13} &\Leftarrow \alpha p_{14} + \sigma p_{16} \\
p_{14} &\Leftarrow \alpha p_{13} + \delta p_{11} + \alpha p_{15} + \mu p_{17} \\
p_{15} &\Leftarrow \theta p_{14} + \sigma p_{12} \\
p_{16} &\Leftarrow \phi p_{17} \\
p_{17} &\Leftarrow \alpha p_{14} + \sigma p_{18} \\
p_{18} &\Leftarrow \phi p_{15} \\
p_{19} &\Leftarrow \psi p_{22} + \xi p_3 \\
p_{20} &\Leftarrow \sigma p_{19} + \alpha p_{23} \\
p_{21} &\Leftarrow \psi p_{20} \\
p_{22} &\Leftarrow \alpha p_{23} + \sigma p_{25} \\
p_{23} &\Leftarrow \alpha p_{22} + \delta p_{20} + \alpha p_{24} + \mu p_{26} \\
p_{24} &\Leftarrow \theta p_{23} + \sigma p_{21} \\
p_{25} &\Leftarrow \psi p_{26} \\
p_{26} &\Leftarrow \alpha p_{23} + \sigma p_{27} \\
p_{27} &\Leftarrow \psi p_{24} \\
p_{28} &\Leftarrow \omega p_{31} + \xi p_7 \\
p_{29} &\Leftarrow \sigma p_{28} + \alpha p_{32} \\
p_{30} &\Leftarrow \omega p_{29} \\
p_{31} &\Leftarrow \alpha p_{32} + \sigma p_{34} \\
p_{32} &\Leftarrow \alpha p_{31} + \delta p_{29} + \alpha p_{33} + \mu p_{35} \\
p_{33} &\Leftarrow \theta p_{32} + \sigma p_{30} \\
p_{34} &\Leftarrow \omega p_{35} \\
p_{35} &\Leftarrow \alpha p_{32} + \sigma p_{36}
\end{aligned}
$$

$$
\begin{aligned}
p_{36} &\Leftarrow \omega p_{33} \\
p_{37} &\Leftarrow \pi p_{40} + \xi p_{9} \\
p_{38} &\Leftarrow \sigma p_{37} + \alpha p_{41} \\
p_{39} &\Leftarrow \pi p_{38} \\
p_{40} &\Leftarrow \alpha p_{41} + \sigma p_{43} \\
p_{41} &\Leftarrow \alpha p_{40} + \delta p_{38} + \alpha p_{42} + \mu p_{44} \\
p_{42} &\Leftarrow \theta p_{41} + \sigma p_{39} \\
p_{43} &\Leftarrow \pi p_{44} \\
p_{44} &\Leftarrow \alpha p_{41} + \sigma p_{45} \\
p_{45} &\Leftarrow \pi p_{42}
\end{aligned}
$$

$\mathcal{M}_q^{540}$ is defined as :-

$$
\begin{aligned}
q_{1} &\Leftarrow \phi q_{4} + \xi q_{11} \\
q_{2} &\Leftarrow \tau q_{50} + \sigma q_{1} \\
q_{3} &\Leftarrow \phi q_{2} \\
q_{4} &\Leftarrow \tau q_{50} + \sigma q_{7} \\
q_{5} &\Leftarrow \tau q_{49} + \tau q_{51} + \tau q_{188} \\
q_{6} &\Leftarrow \sigma q_{3} \\
q_{7} &\Leftarrow \phi q_{8} \\
q_{8} &\Leftarrow \tau q_{50} + \sigma q_{9} \\
q_{9} &\Leftarrow \phi q_{6} \\
q_{10} &\Leftarrow \psi q_{14} + \xi q_{33} \\
q_{11} &\Leftarrow \iota q_{11} + \kappa q_{40} + \xi q_{1} \\
q_{12} &\Leftarrow \tau q_{60} + \sigma q_{10} \\
q_{13} &\Leftarrow \psi q_{12} \\
q_{14} &\Leftarrow \tau q_{60} + \sigma q_{17} \\
q_{15} &\Leftarrow \tau q_{59} + \tau q_{61} + \tau q_{198} \\
q_{16} &\Leftarrow \sigma q_{13} \\
q_{17} &\Leftarrow \psi q_{18} \\
q_{18} &\Leftarrow \tau q_{60} + \sigma q_{19} \\
q_{19} &\Leftarrow \psi q_{16} \\
q_{20} &\Leftarrow \omega q_{24} + \xi q_{43} \\
q_{21} &\Leftarrow \tau q_{70} + \sigma q_{20} \\
q_{22} &\Leftarrow \tau q_{86} + \sigma q_{11} \\
q_{23} &\Leftarrow \omega q_{21} \\
q_{24} &\Leftarrow \tau q_{70} + \sigma q_{27} \\
q_{25} &\Leftarrow \tau q_{69} + \tau q_{71} + \tau q_{208}
\end{aligned}
$$

$$
\begin{aligned}
q_{26} &\Leftarrow \sigma q_{23} \\
q_{27} &\Leftarrow \omega q_{28} \\
q_{28} &\Leftarrow \tau q_{70} + \sigma q_{29} \\
q_{29} &\Leftarrow \omega q_{26} \\
q_{30} &\Leftarrow \pi q_{34} + \xi q_{45} \\
q_{31} &\Leftarrow \tau q_{80} + \sigma q_{30} \\
q_{32} &\Leftarrow \pi q_{31} \\
q_{33} &\Leftarrow \iota q_{33} + \kappa q_{22} + \xi q_{10} \\
q_{34} &\Leftarrow \tau q_{80} + \sigma q_{37} \\
q_{35} &\Leftarrow \tau q_{79} + \tau q_{81} + \tau q_{218} \\
q_{36} &\Leftarrow \sigma q_{32} \\
q_{37} &\Leftarrow \pi q_{38} \\
q_{38} &\Leftarrow \tau q_{80} + \sigma q_{39} \\
q_{39} &\Leftarrow \pi q_{36} \\
q_{40} &\Leftarrow \tau q_{86} + \sigma q_{43} \\
q_{41} &\Leftarrow \tau q_{85} + \tau q_{87} + \tau q_{224} \\
q_{42} &\Leftarrow \sigma q_{33} \\
q_{43} &\Leftarrow \iota q_{43} + \kappa q_{44} + \xi q_{20} \\
q_{44} &\Leftarrow \tau q_{86} + \sigma q_{45} \\
q_{45} &\Leftarrow \iota q_{45} + \kappa q_{42} + \xi q_{30} \\
q_{46} &\Leftarrow \phi q_{49} + \tau q_{181} + \xi q_{56} \\
q_{47} &\Leftarrow \sigma q_{46} + \tau q_{182} \\
q_{48} &\Leftarrow \phi q_{47} + \tau q_{183} \\
q_{49} &\Leftarrow \sigma q_{52} + \tau q_{184} \\
q_{50} &\Leftarrow \tau q_{98} + \tau q_{185} \\
q_{51} &\Leftarrow \sigma q_{48} + \tau q_{186} \\
q_{52} &\Leftarrow \phi q_{53} + \tau q_{187} \\
q_{53} &\Leftarrow \sigma q_{54} + \tau q_{188} \\
q_{54} &\Leftarrow \phi q_{51} + \tau q_{189} \\
q_{55} &\Leftarrow \psi q_{59} + \tau q_{190} + \xi q_{78} \\
q_{56} &\Leftarrow \iota q_{56} + \kappa q_{85} + \tau q_{191} + \xi q_{46} \\
q_{57} &\Leftarrow \sigma q_{55} + \tau q_{192} \\
q_{58} &\Leftarrow \psi q_{57} + \tau q_{193} \\
q_{59} &\Leftarrow \sigma q_{62} + \tau q_{194} \\
q_{60} &\Leftarrow \tau q_{108} + \tau q_{195} \\
q_{61} &\Leftarrow \sigma q_{58} + \tau q_{196} \\
q_{62} &\Leftarrow \psi q_{63} + \tau q_{197} \\
q_{63} &\Leftarrow \sigma q_{64} + \tau q_{198}
\end{aligned}
$$

$$
\begin{aligned}
q_{64} &\Leftarrow \psi q_{61} + \tau q_{199} \\
q_{65} &\Leftarrow \omega q_{69} + \tau q_{200} + \xi q_{88} \\
q_{66} &\Leftarrow \sigma q_{65} + \tau q_{201} \\
q_{67} &\Leftarrow \sigma q_{56} + \tau q_{202} \\
q_{68} &\Leftarrow \omega q_{66} + \tau q_{203} \\
q_{69} &\Leftarrow \sigma q_{71} + \tau q_{204} \\
q_{70} &\Leftarrow \tau q_{118} + \tau q_{205} \\
q_{71} &\Leftarrow \sigma q_{68} + \tau q_{206} \\
q_{72} &\Leftarrow \omega q_{73} + \tau q_{207} \\
q_{73} &\Leftarrow \sigma q_{74} + \tau q_{208} \\
q_{74} &\Leftarrow \omega q_{71} + \tau q_{209} \\
q_{75} &\Leftarrow \pi q_{79} + \tau q_{210} + \xi q_{90} \\
q_{76} &\Leftarrow \sigma q_{75} + \tau q_{211} \\
q_{77} &\Leftarrow \pi q_{76} + \tau q_{212} \\
q_{78} &\Leftarrow \iota q_{78} + \kappa q_{67} + \tau q_{213} + \xi q_{55} \\
q_{79} &\Leftarrow \sigma q_{82} + \tau q_{214} \\
q_{80} &\Leftarrow \tau q_{128} + \tau q_{215} \\
q_{81} &\Leftarrow \sigma q_{77} + \tau q_{216} \\
q_{82} &\Leftarrow \pi q_{83} + \tau q_{217} \\
q_{83} &\Leftarrow \sigma q_{84} + \tau q_{218} \\
q_{84} &\Leftarrow \pi q_{81} + \tau q_{219} \\
q_{85} &\Leftarrow \sigma q_{88} + \tau q_{220} \\
q_{86} &\Leftarrow \tau q_{134} + \tau q_{221} \\
q_{87} &\Leftarrow \sigma q_{78} + \tau q_{222} \\
q_{88} &\Leftarrow \iota q_{88} + \kappa q_{89} + \tau q_{223} + \xi q_{65} \\
q_{89} &\Leftarrow \sigma q_{90} + \tau q_{224} \\
q_{90} &\Leftarrow \iota q_{90} + \kappa q_{87} + \tau q_{225} + \xi q_{75} \\
q_{91} &\Leftarrow \eta q_{136} + \gamma q_{226} + \phi q_{94} + \xi q_{101} \\
q_{92} &\Leftarrow \eta q_{137} + \gamma q_{227} + \sigma q_{91} \\
q_{93} &\Leftarrow \eta q_{138} + \gamma q_{228} + \phi q_{92} \\
q_{94} &\Leftarrow \eta q_{139} + \gamma q_{229} + \sigma q_{97} \\
q_{95} &\Leftarrow \eta q_{140} + \gamma q_{230} \\
q_{96} &\Leftarrow \eta q_{141} + \gamma q_{231} + \sigma q_{93} \\
q_{97} &\Leftarrow \eta q_{142} + \gamma q_{232} + \phi q_{98} \\
q_{98} &\Leftarrow \eta q_{143} + \gamma q_{233} + \sigma q_{99} \\
q_{99} &\Leftarrow \eta q_{144} + \gamma q_{234} + \phi q_{96} \\
q_{100} &\Leftarrow \eta q_{145} + \gamma q_{235} + \psi q_{104} + \xi q_{123} \\
q_{101} &\Leftarrow \eta q_{146} + \gamma q_{236} + \iota q_{101} + \kappa q_{130} + \xi q_{91}
\end{aligned}
$$

$$
\begin{aligned}
q_{102} &\Leftarrow \eta q_{147} + \gamma q_{237} + \sigma q_{100} \\
q_{103} &\Leftarrow \eta q_{148} + \gamma q_{238} + \psi q_{102} \\
q_{104} &\Leftarrow \eta q_{149} + \gamma q_{239} + \sigma q_{107} \\
q_{105} &\Leftarrow \eta q_{150} + \gamma q_{240} \\
q_{106} &\Leftarrow \eta q_{151} + \gamma q_{241} + \sigma q_{103} \\
q_{107} &\Leftarrow \eta q_{152} + \gamma q_{242} + \psi q_{108} \\
q_{108} &\Leftarrow \eta q_{153} + \gamma q_{243} + \sigma q_{109} \\
q_{109} &\Leftarrow \eta q_{154} + \gamma q_{244} + \psi q_{106} \\
q_{110} &\Leftarrow \eta q_{155} + \gamma q_{245} + \omega q_{114} + \xi q_{133} \\
q_{111} &\Leftarrow \eta q_{156} + \gamma q_{246} + \sigma q_{110} \\
q_{112} &\Leftarrow \eta q_{157} + \gamma q_{247} + \sigma q_{101} \\
q_{113} &\Leftarrow \eta q_{158} + \gamma q_{248} + \omega q_{111} \\
q_{114} &\Leftarrow \eta q_{159} + \gamma q_{249} + \sigma q_{117} \\
q_{115} &\Leftarrow \eta q_{160} + \gamma q_{250} \\
q_{116} &\Leftarrow \eta q_{161} + \gamma q_{251} + \sigma q_{113} \\
q_{117} &\Leftarrow \eta q_{162} + \gamma q_{252} + \omega q_{118} \\
q_{118} &\Leftarrow \eta q_{163} + \gamma q_{253} + \sigma q_{119} \\
q_{119} &\Leftarrow \eta q_{164} + \gamma q_{254} + \omega q_{116} \\
q_{120} &\Leftarrow \eta q_{165} + \gamma q_{255} + \pi q_{124} + \xi q_{135} \\
q_{121} &\Leftarrow \eta q_{166} + \gamma q_{256} + \sigma q_{120} \\
q_{122} &\Leftarrow \eta q_{167} + \gamma q_{257} + \pi q_{121} \\
q_{123} &\Leftarrow \eta q_{168} + \gamma q_{258} + \iota q_{123} + \kappa q_{112} + \xi q_{100} \\
q_{124} &\Leftarrow \eta q_{169} + \gamma q_{259} + \sigma q_{127} \\
q_{125} &\Leftarrow \eta q_{170} + \gamma q_{260} \\
q_{126} &\Leftarrow \eta q_{171} + \gamma q_{261} + \sigma q_{122} \\
q_{127} &\Leftarrow \eta q_{172} + \gamma q_{262} + \pi q_{128} \\
q_{128} &\Leftarrow \eta q_{173} + \gamma q_{263} + \sigma q_{129} \\
q_{129} &\Leftarrow \eta q_{174} + \gamma q_{264} + \pi q_{126} \\
q_{130} &\Leftarrow \eta q_{175} + \gamma q_{265} + \sigma q_{133} \\
q_{131} &\Leftarrow \eta q_{176} + \gamma q_{266} \\
q_{132} &\Leftarrow \eta q_{177} + \gamma q_{267} + \sigma q_{123} \\
q_{133} &\Leftarrow \eta q_{178} + \gamma q_{268} + \iota q_{133} + \kappa q_{134} + \xi q_{110} \\
q_{134} &\Leftarrow \eta q_{179} + \gamma q_{269} + \sigma q_{135} \\
q_{135} &\Leftarrow \eta q_{180} + \gamma q_{270} + \iota q_{135} + \kappa q_{132} + \xi q_{120} \\
q_{136} &\Leftarrow \eta q_{316} + \gamma q_{226} + \phi q_{139} + \xi q_{146} \\
q_{137} &\Leftarrow \eta q_{317} + \gamma q_{227} + \sigma q_{136} \\
q_{138} &\Leftarrow \eta q_{318} + \gamma q_{228} + \phi q_{137} \\
q_{139} &\Leftarrow \eta q_{319} + \gamma q_{229} + \sigma q_{142}
\end{aligned}
$$

$$
\begin{aligned}
q_{140} &\Leftarrow \eta q_{320} + \gamma q_{230} \\
q_{141} &\Leftarrow \tau q_{320} + \eta q_{321} + \gamma q_{231} + \sigma q_{138} \\
q_{142} &\Leftarrow \eta q_{322} + \gamma q_{232} + \phi q_{143} \\
q_{143} &\Leftarrow \eta q_{323} + \gamma q_{233} + \sigma q_{144} \\
q_{144} &\Leftarrow \eta q_{324} + \gamma q_{234} + \phi q_{141} \\
q_{145} &\Leftarrow \eta q_{325} + \gamma q_{235} + \psi q_{149} + \xi q_{168} \\
q_{146} &\Leftarrow \eta q_{326} + \gamma q_{236} + \iota q_{146} + \kappa q_{175} + \xi q_{136} \\
q_{147} &\Leftarrow \eta q_{327} + \gamma q_{237} + \sigma q_{145} \\
q_{148} &\Leftarrow \eta q_{328} + \gamma q_{238} + \psi q_{147} \\
q_{149} &\Leftarrow \eta q_{329} + \gamma q_{239} + \sigma q_{152} \\
q_{150} &\Leftarrow \eta q_{330} + \gamma q_{240} \\
q_{151} &\Leftarrow \tau q_{330} + \eta q_{331} + \gamma q_{241} + \sigma q_{148} \\
q_{152} &\Leftarrow \eta q_{332} + \gamma q_{242} + \psi q_{153} \\
q_{153} &\Leftarrow \eta q_{333} + \gamma q_{243} + \sigma q_{154} \\
q_{154} &\Leftarrow \eta q_{334} + \gamma q_{244} + \psi q_{151} \\
q_{155} &\Leftarrow \eta q_{335} + \gamma q_{245} + \omega q_{159} + \xi q_{178} \\
q_{156} &\Leftarrow \eta q_{336} + \gamma q_{246} + \sigma q_{155} \\
q_{157} &\Leftarrow \eta q_{337} + \gamma q_{247} + \sigma q_{146} \\
q_{158} &\Leftarrow \eta q_{338} + \gamma q_{248} + \omega q_{156} \\
q_{159} &\Leftarrow \eta q_{339} + \gamma q_{249} + \sigma q_{162} \\
q_{160} &\Leftarrow \eta q_{340} + \gamma q_{250} \\
q_{161} &\Leftarrow \tau q_{340} + \eta q_{341} + \gamma q_{251} + \sigma q_{158} \\
q_{162} &\Leftarrow \eta q_{342} + \gamma q_{252} + \omega q_{163} \\
q_{163} &\Leftarrow \eta q_{343} + \gamma q_{253} + \sigma q_{164} \\
q_{164} &\Leftarrow \eta q_{344} + \gamma q_{254} + \omega q_{161} \\
q_{165} &\Leftarrow \eta q_{345} + \gamma q_{255} + \pi q_{169} + \xi q_{180} \\
q_{166} &\Leftarrow \eta q_{346} + \gamma q_{256} + \sigma q_{165} \\
q_{167} &\Leftarrow \eta q_{347} + \gamma q_{257} + \pi q_{166} \\
q_{168} &\Leftarrow \eta q_{348} + \gamma q_{258} + \iota q_{168} + \kappa q_{157} + \xi q_{145} \\
q_{169} &\Leftarrow \eta q_{349} + \gamma q_{259} + \sigma q_{172} \\
q_{170} &\Leftarrow \eta q_{350} + \gamma q_{260} \\
q_{171} &\Leftarrow \tau q_{350} + \eta q_{351} + \gamma q_{261} + \sigma q_{167} \\
q_{172} &\Leftarrow \eta q_{352} + \gamma q_{262} + \pi q_{173} \\
q_{173} &\Leftarrow \eta q_{353} + \gamma q_{263} + \sigma q_{174} \\
q_{174} &\Leftarrow \eta q_{354} + \gamma q_{264} + \pi q_{171} \\
q_{175} &\Leftarrow \eta q_{355} + \gamma q_{265} + \sigma q_{178} \\
q_{176} &\Leftarrow \eta q_{356} + \gamma q_{266} \\
q_{177} &\Leftarrow \tau q_{356} + \eta q_{357} + \gamma q_{267} + \sigma q_{168}
\end{aligned}
$$

$$
\begin{aligned}
q_{178} &\Leftarrow \eta q_{358} + \gamma q_{268} + \iota q_{178} + \kappa q_{179} + \xi q_{155} \\
q_{179} &\Leftarrow \eta q_{359} + \gamma q_{269} + \sigma q_{180} \\
q_{180} &\Leftarrow \eta q_{360} + \gamma q_{270} + \iota q_{180} + \kappa q_{177} + \xi q_{165} \\
q_{181} &\Leftarrow \beta q_{181} + \eta q_{1} + \phi q_{184} + \tau q_{46} + \xi q_{191} \\
q_{182} &\Leftarrow \beta q_{182} + \eta q_{2} + \sigma q_{181} + \tau q_{47} \\
q_{183} &\Leftarrow \beta q_{183} + \eta q_{3} + \phi q_{182} + \tau q_{48} \\
q_{184} &\Leftarrow \beta q_{184} + \eta q_{4} + \sigma q_{187} + \tau q_{49} \\
q_{185} &\Leftarrow \tau q_{362} + \tau q_{233} + \beta q_{185} + \eta q_{5} + \tau q_{50} \\
q_{186} &\Leftarrow \tau q_{5} + \beta q_{186} + \eta q_{6} + \sigma q_{183} + \tau q_{51} \\
q_{187} &\Leftarrow \beta q_{187} + \eta q_{7} + \phi q_{188} + \tau q_{52} \\
q_{188} &\Leftarrow \beta q_{188} + \eta q_{8} + \sigma q_{189} + \tau q_{53} \\
q_{189} &\Leftarrow \beta q_{189} + \eta q_{9} + \phi q_{186} + \tau q_{54} \\
q_{190} &\Leftarrow \beta q_{190} + \eta q_{10} + \psi q_{194} + \tau q_{55} + \xi q_{213} \\
q_{191} &\Leftarrow \beta q_{191} + \eta q_{11} + \iota q_{191} + \kappa q_{220} + \tau q_{56} + \xi q_{181} \\
q_{192} &\Leftarrow \beta q_{192} + \eta q_{12} + \sigma q_{190} + \tau q_{57} \\
q_{193} &\Leftarrow \beta q_{193} + \eta q_{13} + \psi q_{192} + \tau q_{58} \\
q_{194} &\Leftarrow \beta q_{194} + \eta q_{14} + \sigma q_{197} + \tau q_{59} \\
q_{195} &\Leftarrow \tau q_{372} + \tau q_{243} + \beta q_{195} + \eta q_{15} + \tau q_{60} \\
q_{196} &\Leftarrow \tau q_{15} + \beta q_{196} + \eta q_{16} + \sigma q_{193} + \tau q_{61} \\
q_{197} &\Leftarrow \beta q_{197} + \eta q_{17} + \psi q_{198} + \tau q_{62} \\
q_{198} &\Leftarrow \beta q_{198} + \eta q_{18} + \sigma q_{199} + \tau q_{63} \\
q_{199} &\Leftarrow \beta q_{199} + \eta q_{19} + \psi q_{196} + \tau q_{64} \\
q_{200} &\Leftarrow \beta q_{200} + \eta q_{20} + \omega q_{204} + \tau q_{65} + \xi q_{223} \\
q_{201} &\Leftarrow \beta q_{201} + \eta q_{21} + \sigma q_{200} + \tau q_{66} \\
q_{202} &\Leftarrow \beta q_{202} + \eta q_{22} + \sigma q_{191} + \tau q_{67} \\
q_{203} &\Leftarrow \beta q_{203} + \eta q_{23} + \omega q_{201} + \tau q_{68} \\
q_{204} &\Leftarrow \beta q_{204} + \eta q_{24} + \sigma q_{207} + \tau q_{69} \\
q_{205} &\Leftarrow \tau q_{381} + \tau q_{253} + \beta q_{205} + \eta q_{25} + \tau q_{70} \\
q_{206} &\Leftarrow \tau q_{25} + \beta q_{206} + \eta q_{26} + \sigma q_{203} + \tau q_{71} \\
q_{207} &\Leftarrow \beta q_{207} + \eta q_{27} + \omega q_{208} + \tau q_{72} \\
q_{208} &\Leftarrow \beta q_{208} + \eta q_{28} + \sigma q_{209} + \tau q_{73} \\
q_{209} &\Leftarrow \beta q_{209} + \eta q_{29} + \omega q_{206} + \tau q_{74} \\
q_{210} &\Leftarrow \beta q_{210} + \eta q_{30} + \pi q_{214} + \tau q_{75} + \xi q_{225} \\
q_{211} &\Leftarrow \beta q_{211} + \eta q_{31} + \sigma q_{210} + \tau q_{76} \\
q_{212} &\Leftarrow \beta q_{212} + \eta q_{32} + \pi q_{211} + \tau q_{77} \\
q_{213} &\Leftarrow \beta q_{213} + \eta q_{33} + \iota q_{213} + \kappa q_{202} + \tau q_{78} + \xi q_{190} \\
q_{214} &\Leftarrow \beta q_{214} + \eta q_{34} + \sigma q_{217} + \tau q_{79} \\
q_{215} &\Leftarrow \tau q_{391} + \tau q_{263} + \beta q_{215} + \eta q_{35} + \tau q_{80}
\end{aligned}
$$

$$
\begin{aligned}
q_{216} &\Leftarrow \tau q_{35} + \beta q_{216} + \eta q_{36} + \sigma q_{212} + \tau q_{81} \\
q_{217} &\Leftarrow \beta q_{217} + \eta q_{37} + \pi q_{218} + \tau q_{82} \\
q_{218} &\Leftarrow \beta q_{218} + \eta q_{38} + \sigma q_{219} + \tau q_{83} \\
q_{219} &\Leftarrow \beta q_{219} + \eta q_{39} + \pi q_{216} + \tau q_{84} \\
q_{220} &\Leftarrow \beta q_{220} + \eta q_{40} + \sigma q_{223} + \tau q_{85} \\
q_{221} &\Leftarrow \tau q_{382} + \tau q_{269} + \beta q_{221} + \eta q_{41} + \tau q_{86} \\
q_{222} &\Leftarrow \tau q_{41} + \beta q_{222} + \eta q_{42} + \sigma q_{213} + \tau q_{87} \\
q_{223} &\Leftarrow \beta q_{223} + \eta q_{43} + \iota q_{223} + \kappa q_{224} + \tau q_{88} + \xi q_{200} \\
q_{224} &\Leftarrow \beta q_{224} + \eta q_{44} + \sigma q_{225} + \tau q_{89} \\
q_{225} &\Leftarrow \beta q_{225} + \eta q_{45} + \iota q_{225} + \kappa q_{222} + \tau q_{90} + \xi q_{210} \\
q_{226} &\Leftarrow \beta q_{46} + \gamma q_{361} + \phi q_{229} + \xi q_{236} \\
q_{227} &\Leftarrow \tau q_{275} + \beta q_{47} + \gamma q_{362} + \sigma q_{226} \\
q_{228} &\Leftarrow \beta q_{48} + \gamma q_{363} + \phi q_{227} \\
q_{229} &\Leftarrow \tau q_{275} + \beta q_{49} + \gamma q_{364} + \sigma q_{232} \\
q_{230} &\Leftarrow \tau q_{92} + \tau q_{274} + \tau q_{276} + \beta q_{50} + \gamma q_{365} \\
q_{231} &\Leftarrow \tau q_{185} + \beta q_{51} + \gamma q_{366} + \sigma q_{228} \\
q_{232} &\Leftarrow \beta q_{52} + \gamma q_{367} + \phi q_{233} \\
q_{233} &\Leftarrow \tau q_{275} + \beta q_{53} + \gamma q_{368} + \sigma q_{234} \\
q_{234} &\Leftarrow \beta q_{54} + \gamma q_{369} + \phi q_{231} \\
q_{235} &\Leftarrow \beta q_{55} + \gamma q_{370} + \psi q_{239} + \xi q_{258} \\
q_{236} &\Leftarrow \beta q_{56} + \gamma q_{371} + \iota q_{236} + \kappa q_{265} + \xi q_{226} \\
q_{237} &\Leftarrow \tau q_{285} + \beta q_{57} + \gamma q_{372} + \sigma q_{235} \\
q_{238} &\Leftarrow \beta q_{58} + \gamma q_{373} + \psi q_{237} \\
q_{239} &\Leftarrow \tau q_{285} + \beta q_{59} + \gamma q_{374} + \sigma q_{242} \\
q_{240} &\Leftarrow \tau q_{102} + \tau q_{284} + \tau q_{286} + \beta q_{60} + \gamma q_{375} \\
q_{241} &\Leftarrow \tau q_{195} + \beta q_{61} + \gamma q_{376} + \sigma q_{238} \\
q_{242} &\Leftarrow \beta q_{62} + \gamma q_{377} + \psi q_{243} \\
q_{243} &\Leftarrow \tau q_{285} + \beta q_{63} + \gamma q_{378} + \sigma q_{244} \\
q_{244} &\Leftarrow \beta q_{64} + \gamma q_{379} + \psi q_{241} \\
q_{245} &\Leftarrow \beta q_{65} + \gamma q_{380} + \omega q_{249} + \xi q_{268} \\
q_{246} &\Leftarrow \tau q_{295} + \beta q_{66} + \gamma q_{381} + \sigma q_{245} \\
q_{247} &\Leftarrow \tau q_{311} + \beta q_{67} + \gamma q_{382} + \sigma q_{236} \\
q_{248} &\Leftarrow \beta q_{68} + \gamma q_{383} + \omega q_{246} \\
q_{249} &\Leftarrow \tau q_{295} + \beta q_{69} + \gamma q_{384} + \sigma q_{252} \\
q_{250} &\Leftarrow \tau q_{111} + \tau q_{294} + \tau q_{296} + \beta q_{70} + \gamma q_{385} \\
q_{251} &\Leftarrow \tau q_{205} + \beta q_{71} + \gamma q_{386} + \sigma q_{248} \\
q_{252} &\Leftarrow \beta q_{72} + \gamma q_{387} + \omega q_{253} \\
q_{253} &\Leftarrow \tau q_{295} + \beta q_{73} + \gamma q_{388} + \sigma q_{254}
\end{aligned}
$$

$$
\begin{aligned}
q_{254} &\Leftarrow \beta q_{74} + \gamma q_{389} + \omega q_{251} \\
q_{255} &\Leftarrow \beta q_{75} + \gamma q_{390} + \pi q_{259} + \xi q_{270} \\
q_{256} &\Leftarrow \tau q_{305} + \beta q_{76} + \gamma q_{391} + \sigma q_{255} \\
q_{257} &\Leftarrow \beta q_{77} + \gamma q_{392} + \pi q_{256} \\
q_{258} &\Leftarrow \beta q_{78} + \gamma q_{393} + \iota q_{258} + \kappa q_{247} + \xi q_{235} \\
q_{259} &\Leftarrow \tau q_{305} + \beta q_{79} + \gamma q_{394} + \sigma q_{262} \\
q_{260} &\Leftarrow \tau q_{121} + \tau q_{304} + \tau q_{306} + \beta q_{80} + \gamma q_{395} \\
q_{261} &\Leftarrow \tau q_{215} + \beta q_{81} + \gamma q_{396} + \sigma q_{257} \\
q_{262} &\Leftarrow \beta q_{82} + \gamma q_{397} + \pi q_{263} \\
q_{263} &\Leftarrow \tau q_{305} + \beta q_{83} + \gamma q_{398} + \sigma q_{264} \\
q_{264} &\Leftarrow \beta q_{84} + \gamma q_{399} + \pi q_{261} \\
q_{265} &\Leftarrow \tau q_{311} + \beta q_{85} + \gamma q_{400} + \sigma q_{268} \\
q_{266} &\Leftarrow \tau q_{112} + \tau q_{310} + \tau q_{312} + \beta q_{86} + \gamma q_{401} \\
q_{267} &\Leftarrow \tau q_{221} + \beta q_{87} + \gamma q_{402} + \sigma q_{258} \\
q_{268} &\Leftarrow \beta q_{88} + \gamma q_{403} + \iota q_{268} + \kappa q_{269} + \xi q_{245} \\
q_{269} &\Leftarrow \tau q_{311} + \beta q_{89} + \gamma q_{404} + \sigma q_{270} \\
q_{270} &\Leftarrow \beta q_{90} + \gamma q_{405} + \iota q_{270} + \kappa q_{267} + \xi q_{255} \\
q_{271} &\Leftarrow \phi q_{274} + \tau q_{406} + \xi q_{281} \\
q_{272} &\Leftarrow \sigma q_{271} + \tau q_{407} \\
q_{273} &\Leftarrow \phi q_{272} + \tau q_{408} \\
q_{274} &\Leftarrow \sigma q_{277} + \tau q_{409} \\
q_{275} &\Leftarrow \tau q_{503} + \tau q_{410} \\
q_{276} &\Leftarrow \sigma q_{273} + \tau q_{411} \\
q_{277} &\Leftarrow \phi q_{278} + \tau q_{412} \\
q_{278} &\Leftarrow \sigma q_{279} + \tau q_{413} \\
q_{279} &\Leftarrow \phi q_{276} + \tau q_{414} \\
q_{280} &\Leftarrow \psi q_{284} + \tau q_{415} + \xi q_{303} \\
q_{281} &\Leftarrow \iota q_{281} + \kappa q_{310} + \tau q_{416} + \xi q_{271} \\
q_{282} &\Leftarrow \sigma q_{280} + \tau q_{417} \\
q_{283} &\Leftarrow \psi q_{282} + \tau q_{418} \\
q_{284} &\Leftarrow \sigma q_{287} + \tau q_{419} \\
q_{285} &\Leftarrow \tau q_{513} + \tau q_{420} \\
q_{286} &\Leftarrow \sigma q_{283} + \tau q_{421} \\
q_{287} &\Leftarrow \psi q_{288} + \tau q_{422} \\
q_{288} &\Leftarrow \sigma q_{289} + \tau q_{423} \\
q_{289} &\Leftarrow \psi q_{286} + \tau q_{424} \\
q_{290} &\Leftarrow \omega q_{294} + \tau q_{425} + \xi q_{313} \\
q_{291} &\Leftarrow \sigma q_{290} + \tau q_{426}
\end{aligned}
$$

$$
\begin{aligned}
q_{292} &\Leftarrow \sigma q_{281} + \tau q_{427} \\
q_{293} &\Leftarrow \omega q_{291} + \tau q_{428} \\
q_{294} &\Leftarrow \sigma q_{297} + \tau q_{429} \\
q_{295} &\Leftarrow \tau q_{523} + \tau q_{430} \\
q_{296} &\Leftarrow \sigma q_{293} + \tau q_{431} \\
q_{297} &\Leftarrow \omega q_{298} + \tau q_{432} \\
q_{298} &\Leftarrow \sigma q_{299} + \tau q_{433} \\
q_{299} &\Leftarrow \omega q_{296} + \tau q_{434} \\
q_{300} &\Leftarrow \pi q_{304} + \tau q_{435} + \xi q_{315} \\
q_{301} &\Leftarrow \sigma q_{300} + \tau q_{436} \\
q_{302} &\Leftarrow \pi q_{301} + \tau q_{437} \\
q_{303} &\Leftarrow \iota q_{303} + \kappa q_{292} + \tau q_{438} + \xi q_{280} \\
q_{304} &\Leftarrow \sigma q_{307} + \tau q_{439} \\
q_{305} &\Leftarrow \tau q_{533} + \tau q_{440} \\
q_{306} &\Leftarrow \sigma q_{302} + \tau q_{441} \\
q_{307} &\Leftarrow \pi q_{308} + \tau q_{442} \\
q_{308} &\Leftarrow \sigma q_{309} + \tau q_{443} \\
q_{309} &\Leftarrow \pi q_{306} + \tau q_{444} \\
q_{310} &\Leftarrow \sigma q_{313} + \tau q_{445} \\
q_{311} &\Leftarrow \tau q_{539} + \tau q_{446} \\
q_{312} &\Leftarrow \sigma q_{303} + \tau q_{447} \\
q_{313} &\Leftarrow \iota q_{313} + \kappa q_{314} + \tau q_{448} + \xi q_{290} \\
q_{314} &\Leftarrow \sigma q_{315} + \tau q_{449} \\
q_{315} &\Leftarrow \iota q_{315} + \kappa q_{312} + \tau q_{450} + \xi q_{300} \\
q_{316} &\Leftarrow \phi q_{319} + \xi q_{326} \\
q_{317} &\Leftarrow \tau q_{320} + \sigma q_{316} \\
q_{318} &\Leftarrow \phi q_{317} \\
q_{319} &\Leftarrow \tau q_{320} + \sigma q_{322} \\
q_{320} &\Leftarrow \tau q_{497} + \tau q_{319} + \tau q_{321} + \tau q_{143} \\
q_{321} &\Leftarrow \sigma q_{318} \\
q_{322} &\Leftarrow \phi q_{323} \\
q_{323} &\Leftarrow \tau q_{320} + \sigma q_{324} \\
q_{324} &\Leftarrow \phi q_{321} \\
q_{325} &\Leftarrow \psi q_{329} + \xi q_{348} \\
q_{326} &\Leftarrow \iota q_{326} + \kappa q_{355} + \xi q_{316} \\
q_{327} &\Leftarrow \tau q_{330} + \sigma q_{325} \\
q_{328} &\Leftarrow \psi q_{327} \\
q_{329} &\Leftarrow \tau q_{330} + \sigma q_{332}
\end{aligned}
$$

$$
\begin{aligned}
q_{330} &\Leftarrow \tau q_{507} + \tau q_{329} + \tau q_{331} + \tau q_{153} \\
q_{331} &\Leftarrow \sigma q_{328} \\
q_{332} &\Leftarrow \psi q_{333} \\
q_{333} &\Leftarrow \tau q_{330} + \sigma q_{334} \\
q_{334} &\Leftarrow \psi q_{331} \\
q_{335} &\Leftarrow \omega q_{339} + \xi q_{358} \\
q_{336} &\Leftarrow \tau q_{340} + \sigma q_{335} \\
q_{337} &\Leftarrow \tau q_{356} + \sigma q_{326} \\
q_{338} &\Leftarrow \omega q_{336} \\
q_{339} &\Leftarrow \tau q_{340} + \sigma q_{342} \\
q_{340} &\Leftarrow \tau q_{516} + \tau q_{339} + \tau q_{341} + \tau q_{163} \\
q_{341} &\Leftarrow \sigma q_{338} \\
q_{342} &\Leftarrow \omega q_{343} \\
q_{343} &\Leftarrow \tau q_{340} + \sigma q_{344} \\
q_{344} &\Leftarrow \omega q_{341} \\
q_{345} &\Leftarrow \pi q_{349} + \xi q_{360} \\
q_{346} &\Leftarrow \tau q_{350} + \sigma q_{345} \\
q_{347} &\Leftarrow \pi q_{346} \\
q_{348} &\Leftarrow \iota q_{348} + \kappa q_{337} + \xi q_{325} \\
q_{349} &\Leftarrow \tau q_{350} + \sigma q_{352} \\
q_{350} &\Leftarrow \tau q_{526} + \tau q_{349} + \tau q_{351} + \tau q_{173} \\
q_{351} &\Leftarrow \sigma q_{347} \\
q_{352} &\Leftarrow \pi q_{353} \\
q_{353} &\Leftarrow \tau q_{350} + \sigma q_{354} \\
q_{354} &\Leftarrow \pi q_{351} \\
q_{355} &\Leftarrow \tau q_{356} + \sigma q_{358} \\
q_{356} &\Leftarrow \tau q_{517} + \tau q_{355} + \tau q_{357} + \tau q_{179} \\
q_{357} &\Leftarrow \sigma q_{348} \\
q_{358} &\Leftarrow \iota q_{358} + \kappa q_{359} + \xi q_{335} \\
q_{359} &\Leftarrow \tau q_{356} + \sigma q_{360} \\
q_{360} &\Leftarrow \iota q_{360} + \kappa q_{357} + \xi q_{345} \\
q_{361} &\Leftarrow \gamma q_{181} + \phi q_{364} + \xi q_{371} \\
q_{362} &\Leftarrow \gamma q_{182} + \sigma q_{361} \\
q_{363} &\Leftarrow \gamma q_{183} + \phi q_{362} \\
q_{364} &\Leftarrow \gamma q_{184} + \sigma q_{367} \\
q_{365} &\Leftarrow \tau q_{227} + \gamma q_{185} \\
q_{366} &\Leftarrow \gamma q_{186} + \sigma q_{363} \\
q_{367} &\Leftarrow \gamma q_{187} + \phi q_{368}
\end{aligned}
$$

$$
\begin{aligned}
q_{368} &\Leftarrow \gamma q_{188} + \sigma q_{369} \\
q_{369} &\Leftarrow \gamma q_{189} + \phi q_{366} \\
q_{370} &\Leftarrow \gamma q_{190} + \psi q_{374} + \xi q_{393} \\
q_{371} &\Leftarrow \gamma q_{191} + \iota q_{371} + \kappa q_{400} + \xi q_{361} \\
q_{372} &\Leftarrow \gamma q_{192} + \sigma q_{370} \\
q_{373} &\Leftarrow \gamma q_{193} + \psi q_{372} \\
q_{374} &\Leftarrow \gamma q_{194} + \sigma q_{377} \\
q_{375} &\Leftarrow \tau q_{237} + \gamma q_{195} \\
q_{376} &\Leftarrow \gamma q_{196} + \sigma q_{373} \\
q_{377} &\Leftarrow \gamma q_{197} + \psi q_{378} \\
q_{378} &\Leftarrow \gamma q_{198} + \sigma q_{379} \\
q_{379} &\Leftarrow \gamma q_{199} + \psi q_{376} \\
q_{380} &\Leftarrow \gamma q_{200} + \omega q_{384} + \xi q_{403} \\
q_{381} &\Leftarrow \gamma q_{201} + \sigma q_{380} \\
q_{382} &\Leftarrow \gamma q_{202} + \sigma q_{371} \\
q_{383} &\Leftarrow \gamma q_{203} + \omega q_{381} \\
q_{384} &\Leftarrow \gamma q_{204} + \sigma q_{387} \\
q_{385} &\Leftarrow \tau q_{246} + \gamma q_{205} \\
q_{386} &\Leftarrow \gamma q_{206} + \sigma q_{383} \\
q_{387} &\Leftarrow \gamma q_{207} + \omega q_{388} \\
q_{388} &\Leftarrow \gamma q_{208} + \sigma q_{389} \\
q_{389} &\Leftarrow \gamma q_{209} + \omega q_{386} \\
q_{390} &\Leftarrow \gamma q_{210} + \pi q_{394} + \xi q_{405} \\
q_{391} &\Leftarrow \gamma q_{211} + \sigma q_{390} \\
q_{392} &\Leftarrow \gamma q_{212} + \pi q_{391} \\
q_{393} &\Leftarrow \gamma q_{213} + \iota q_{393} + \kappa q_{382} + \xi q_{370} \\
q_{394} &\Leftarrow \gamma q_{214} + \sigma q_{397} \\
q_{395} &\Leftarrow \tau q_{256} + \gamma q_{215} \\
q_{396} &\Leftarrow \gamma q_{216} + \sigma q_{393} \\
q_{397} &\Leftarrow \gamma q_{217} + \pi q_{398} \\
q_{398} &\Leftarrow \gamma q_{218} + \sigma q_{399} \\
q_{399} &\Leftarrow \gamma q_{219} + \pi q_{396} \\
q_{400} &\Leftarrow \gamma q_{220} + \sigma q_{403} \\
q_{401} &\Leftarrow \tau q_{247} + \gamma q_{221} \\
q_{402} &\Leftarrow \gamma q_{222} + \sigma q_{392} \\
q_{403} &\Leftarrow \gamma q_{223} + \iota q_{403} + \kappa q_{404} + \xi q_{380} \\
q_{404} &\Leftarrow \gamma q_{224} + \sigma q_{405} \\
q_{405} &\Leftarrow \gamma q_{225} + \iota q_{405} + \kappa q_{402} + \xi q_{390}
\end{aligned}
$$

$$
\begin{aligned}
q_{406} &\Leftarrow \beta q_{226} + \phi q_{409} + \tau q_{271} + \xi q_{416} \\
q_{407} &\Leftarrow \beta q_{227} + \sigma q_{10} + \tau q_{272} \\
q_{408} &\Leftarrow \beta q_{228} + \phi q_{407} + \tau q_{273} \\
q_{409} &\Leftarrow \beta q_{229} + \sigma q_{412} + \tau q_{274} \\
q_{410} &\Leftarrow \tau q_{452} + \beta q_{230} + \tau q_{275} \\
q_{411} &\Leftarrow \beta q_{231} + \sigma q_{408} + \tau q_{276} \\
q_{412} &\Leftarrow \beta q_{232} + \phi q_{413} + \tau q_{277} \\
q_{413} &\Leftarrow \beta q_{233} + \sigma q_{414} + \tau q_{278} \\
q_{414} &\Leftarrow \beta q_{234} + \phi q_{411} + \tau q_{279} \\
q_{415} &\Leftarrow \beta q_{235} + \psi q_{419} + \tau q_{280} + \xi q_{438} \\
q_{416} &\Leftarrow \beta q_{236} + \iota q_{416} + \kappa q_{445} + \tau q_{281} + \xi q_{406} \\
q_{417} &\Leftarrow \beta q_{237} + \sigma q_{415} + \tau q_{282} \\
q_{418} &\Leftarrow \beta q_{238} + \psi q_{417} + \tau q_{283} \\
q_{419} &\Leftarrow \beta q_{239} + \sigma q_{422} + \tau q_{284} \\
q_{420} &\Leftarrow \tau q_{462} + \beta q_{240} + \tau q_{285} \\
q_{421} &\Leftarrow \beta q_{241} + \sigma q_{418} + \tau q_{286} \\
q_{422} &\Leftarrow \beta q_{242} + \psi q_{423} + \tau q_{287} \\
q_{423} &\Leftarrow \beta q_{243} + \sigma q_{424} + \tau q_{288} \\
q_{424} &\Leftarrow \beta q_{244} + \psi q_{421} + \tau q_{289} \\
q_{425} &\Leftarrow \beta q_{245} + \omega q_{429} + \tau q_{290} + \xi q_{448} \\
q_{426} &\Leftarrow \beta q_{246} + \sigma q_{425} + \tau q_{291} \\
q_{427} &\Leftarrow \beta q_{247} + \sigma q_{416} + \tau q_{292} \\
q_{428} &\Leftarrow \beta q_{248} + \omega q_{426} + \tau q_{293} \\
q_{429} &\Leftarrow \beta q_{249} + \sigma q_{432} + \tau q_{294} \\
q_{430} &\Leftarrow \tau q_{471} + \beta q_{250} + \tau q_{295} \\
q_{431} &\Leftarrow \beta q_{251} + \sigma q_{428} + \tau q_{296} \\
q_{432} &\Leftarrow \beta q_{252} + \omega q_{433} + \tau q_{297} \\
q_{433} &\Leftarrow \beta q_{253} + \sigma q_{434} + \tau q_{298} \\
q_{434} &\Leftarrow \beta q_{254} + \omega q_{431} + \tau q_{299} \\
q_{435} &\Leftarrow \beta q_{255} + \pi q_{439} + \tau q_{300} + \xi q_{450} \\
q_{436} &\Leftarrow \beta q_{256} + \sigma q_{435} + \tau q_{301} \\
q_{437} &\Leftarrow \beta q_{257} + \pi q_{436} + \tau q_{302} \\
q_{438} &\Leftarrow \beta q_{258} + \iota q_{438} + \kappa q_{427} + \tau q_{303} + \xi q_{415} \\
q_{439} &\Leftarrow \beta q_{259} + \sigma q_{442} + \tau q_{304} \\
q_{440} &\Leftarrow \tau q_{481} + \beta q_{260} + \tau q_{305} \\
q_{441} &\Leftarrow \beta q_{261} + \sigma q_{437} + \tau q_{306} \\
q_{442} &\Leftarrow \beta q_{262} + \pi q_{443} + \tau q_{307} \\
q_{443} &\Leftarrow \beta q_{263} + \sigma q_{444} + \tau q_{308}
\end{aligned}
$$

$$
\begin{aligned}
q_{444} &\Leftarrow \beta q_{264} + \pi q_{441} + \tau q_{309} \\
q_{445} &\Leftarrow \beta q_{265} + \sigma q_{448} + \tau q_{310} \\
q_{446} &\Leftarrow \tau q_{472} + \beta q_{266} + \tau q_{311} \\
q_{447} &\Leftarrow \beta q_{267} + \sigma q_{438} + \tau q_{312} \\
q_{448} &\Leftarrow \beta q_{268} + \iota q_{448} + \kappa q_{449} + \tau q_{313} + \xi q_{425} \\
q_{449} &\Leftarrow \beta q_{269} + \sigma q_{450} + \tau q_{314} \\
q_{450} &\Leftarrow \beta q_{270} + \iota q_{450} + \kappa q_{447} + \tau q_{315} + \xi q_{435} \\
q_{451} &\Leftarrow \gamma q_{406} + \phi q_{454} + \xi q_{461} \\
q_{452} &\Leftarrow \gamma q_{407} + \sigma q_{1} \\
q_{453} &\Leftarrow \gamma q_{408} + \phi q_{452} \\
q_{454} &\Leftarrow \gamma q_{409} + \sigma q_{457} \\
q_{455} &\Leftarrow \gamma q_{410} \\
q_{456} &\Leftarrow \gamma q_{411} + \sigma q_{453} \\
q_{457} &\Leftarrow \gamma q_{412} + \phi q_{458} \\
q_{458} &\Leftarrow \gamma q_{413} + \sigma q_{459} \\
q_{459} &\Leftarrow \gamma q_{414} + \phi q_{456} \\
q_{460} &\Leftarrow \gamma q_{415} + \psi q_{464} + \xi q_{483} \\
q_{461} &\Leftarrow \gamma q_{416} + \iota q_{461} + \kappa q_{490} + \xi q_{451} \\
q_{462} &\Leftarrow \gamma q_{417} + \sigma q_{460} \\
q_{463} &\Leftarrow \gamma q_{418} + \psi q_{462} \\
q_{464} &\Leftarrow \gamma q_{419} + \sigma q_{467} \\
q_{465} &\Leftarrow \gamma q_{420} \\
q_{466} &\Leftarrow \gamma q_{421} + \sigma q_{463} \\
q_{467} &\Leftarrow \gamma q_{422} + \psi q_{468} \\
q_{468} &\Leftarrow \gamma q_{423} + \sigma q_{469} \\
q_{469} &\Leftarrow \gamma q_{424} + \psi q_{466} \\
q_{470} &\Leftarrow \gamma q_{425} + \omega q_{474} + \xi q_{493} \\
q_{471} &\Leftarrow \gamma q_{426} + \sigma q_{470} \\
q_{472} &\Leftarrow \gamma q_{427} + \sigma q_{461} \\
q_{473} &\Leftarrow \gamma q_{428} + \omega q_{471} \\
q_{474} &\Leftarrow \gamma q_{429} + \sigma q_{477} \\
q_{475} &\Leftarrow \gamma q_{430} \\
q_{476} &\Leftarrow \gamma q_{431} + \sigma q_{473} \\
q_{477} &\Leftarrow \gamma q_{432} + \omega q_{478} \\
q_{478} &\Leftarrow \gamma q_{433} + \sigma q_{479} \\
q_{479} &\Leftarrow \gamma q_{434} + \omega q_{476} \\
q_{480} &\Leftarrow \gamma q_{435} + \pi q_{484} + \xi q_{495} \\
q_{481} &\Leftarrow \gamma q_{436} + \sigma q_{480}
\end{aligned}
$$

$$
\begin{aligned}
q_{482} &\Leftarrow \gamma q_{437} + \pi q_{481} \\
q_{483} &\Leftarrow \gamma q_{438} + \iota q_{483} + \kappa q_{472} + \xi q_{460} \\
q_{484} &\Leftarrow \gamma q_{439} + \sigma q_{487} \\
q_{485} &\Leftarrow \gamma q_{440} \\
q_{486} &\Leftarrow \gamma q_{441} + \sigma q_{482} \\
q_{487} &\Leftarrow \gamma q_{442} + \pi q_{488} \\
q_{488} &\Leftarrow \gamma q_{443} + \sigma q_{489} \\
q_{489} &\Leftarrow \gamma q_{444} + \pi q_{486} \\
q_{490} &\Leftarrow \gamma q_{445} + \sigma q_{493} \\
q_{491} &\Leftarrow \gamma q_{446} \\
q_{492} &\Leftarrow \gamma q_{447} + \sigma q_{483} \\
q_{493} &\Leftarrow \gamma q_{448} + \iota q_{493} + \kappa q_{494} + \xi q_{470} \\
q_{494} &\Leftarrow \gamma q_{449} + \sigma q_{495} \\
q_{495} &\Leftarrow \gamma q_{450} + \iota q_{495} + \kappa q_{492} + \xi q_{480} \\
q_{496} &\Leftarrow \eta q_{271} + \gamma q_{316} + \phi q_{499} + \xi q_{506} \\
q_{497} &\Leftarrow \eta q_{272} + \gamma q_{317} + \sigma q_{496} \\
q_{498} &\Leftarrow \eta q_{273} + \gamma q_{318} + \phi q_{497} \\
q_{499} &\Leftarrow \eta q_{274} + \gamma q_{319} + \sigma q_{502} \\
q_{500} &\Leftarrow \eta q_{275} + \gamma q_{320} \\
q_{501} &\Leftarrow \tau q_{275} + \eta q_{276} + \gamma q_{321} + \sigma q_{498} \\
q_{502} &\Leftarrow \eta q_{277} + \gamma q_{322} + \phi q_{503} \\
q_{503} &\Leftarrow \eta q_{278} + \gamma q_{323} + \sigma q_{504} \\
q_{504} &\Leftarrow \eta q_{279} + \gamma q_{324} + \phi q_{501} \\
q_{505} &\Leftarrow \eta q_{280} + \gamma q_{325} + \psi q_{509} + \xi q_{528} \\
q_{506} &\Leftarrow \eta q_{281} + \gamma q_{326} + \iota q_{506} + \kappa q_{535} + \xi q_{496} \\
q_{507} &\Leftarrow \eta q_{282} + \gamma q_{327} + \sigma q_{505} \\
q_{508} &\Leftarrow \eta q_{283} + \gamma q_{328} + \psi q_{507} \\
q_{509} &\Leftarrow \eta q_{284} + \gamma q_{329} + \sigma q_{512} \\
q_{510} &\Leftarrow \eta q_{285} + \gamma q_{330} \\
q_{511} &\Leftarrow \tau q_{285} + \eta q_{286} + \gamma q_{331} + \sigma q_{508} \\
q_{512} &\Leftarrow \eta q_{287} + \gamma q_{332} + \psi q_{513} \\
q_{513} &\Leftarrow \eta q_{288} + \gamma q_{333} + \sigma q_{514} \\
q_{514} &\Leftarrow \eta q_{289} + \gamma q_{334} + \psi q_{511} \\
q_{515} &\Leftarrow \eta q_{290} + \gamma q_{335} + \omega q_{519} + \xi q_{538} \\
q_{516} &\Leftarrow \eta q_{291} + \gamma q_{336} + \sigma q_{515} \\
q_{517} &\Leftarrow \eta q_{292} + \gamma q_{337} + \sigma q_{506} \\
q_{518} &\Leftarrow \eta q_{293} + \gamma q_{338} + \omega q_{516} \\
q_{519} &\Leftarrow \eta q_{294} + \gamma q_{339} + \sigma q_{522}
\end{aligned}
$$

$$
\begin{aligned}
q_{520} &\Leftarrow \eta q_{295} + \gamma q_{340} \\
q_{521} &\Leftarrow \tau q_{295} + \eta q_{296} + \gamma q_{341} + \sigma q_{518} \\
q_{522} &\Leftarrow \eta q_{297} + \gamma q_{342} + \omega q_{523} \\
q_{523} &\Leftarrow \eta q_{298} + \gamma q_{343} + \sigma q_{525} \\
q_{525} &\Leftarrow \eta q_{299} + \gamma q_{344} + \omega q_{521} \\
q_{524} &\Leftarrow \eta q_{300} + \gamma q_{345} + \pi q_{529} + \xi q_{540} \\
q_{526} &\Leftarrow \eta q_{301} + \gamma q_{346} + \sigma q_{524} \\
q_{527} &\Leftarrow \eta q_{302} + \gamma q_{347} + \pi q_{526} \\
q_{528} &\Leftarrow \eta q_{303} + \gamma q_{348} + \iota q_{528} + \kappa q_{517} + \xi q_{505} \\
q_{529} &\Leftarrow \eta q_{304} + \gamma q_{349} + \sigma q_{532} \\
q_{530} &\Leftarrow \eta q_{305} + \gamma q_{350} \\
q_{531} &\Leftarrow \tau q_{305} + \eta q_{306} + \gamma q_{351} + \sigma q_{527} \\
q_{532} &\Leftarrow \eta q_{307} + \gamma q_{352} + \pi q_{533} \\
q_{533} &\Leftarrow \eta q_{308} + \gamma q_{353} + \sigma q_{534} \\
q_{534} &\Leftarrow \eta q_{309} + \gamma q_{354} + \pi q_{531} \\
q_{535} &\Leftarrow \eta q_{310} + \gamma q_{355} + \sigma q_{538} \\
q_{536} &\Leftarrow \eta q_{311} + \gamma q_{356} \\
q_{537} &\Leftarrow \tau q_{311} + \eta q_{312} + \gamma q_{357} + \sigma q_{528} \\
q_{538} &\Leftarrow \eta q_{313} + \gamma q_{358} + \iota q_{538} + \kappa q_{539} + \xi q_{515} \\
q_{539} &\Leftarrow \eta q_{314} + \gamma q_{359} + \sigma q_{540} \\
q_{540} &\Leftarrow \eta q_{315} + \gamma q_{360} + \iota q_{540} + \kappa q_{537} + \xi q_{524}
\end{aligned}
$$

Removal of all non-*tau* actions leaves :-

$$
\begin{aligned}
q_{2} &\Leftarrow \tau_c q_{50} \\
q_{4} &\Leftarrow \tau_c q_{50} \\
q_{5} &\Leftarrow \tau_c q_{49} + \tau_c q_{51} + \tau_c q_{188} \\
q_{8} &\Leftarrow \tau_c q_{50} \\
q_{12} &\Leftarrow \tau_c q_{60} \\
q_{14} &\Leftarrow \tau_c q_{60} \\
q_{15} &\Leftarrow \tau_c q_{59} + \tau_c q_{61} + \tau_c q_{198} \\
q_{18} &\Leftarrow \tau_c q_{60} \\
q_{21} &\Leftarrow \tau_c q_{70} \\
q_{22} &\Leftarrow \tau_c q_{86} \\
q_{24} &\Leftarrow \tau_c q_{70} \\
q_{25} &\Leftarrow \tau_c q_{69} + \tau_c q_{71} + \tau_c q_{208} \\
q_{28} &\Leftarrow \tau_c q_{70} \\
q_{31} &\Leftarrow \tau_c q_{80} \\
q_{34} &\Leftarrow \tau_c q_{80} \\
q_{35} &\Leftarrow \tau_c q_{79} + \tau_c q_{81} + \tau_c q_{218} \\
q_{38} &\Leftarrow \tau_c q_{80} \\
q_{40} &\Leftarrow \tau_c q_{86} \\
q_{41} &\Leftarrow \tau_c q_{85} + \tau_c q_{87} + \tau_c q_{224} \\
q_{44} &\Leftarrow \tau_c q_{86} \\
q_{46} &\Leftarrow \tau q_{181} \\
q_{47} &\Leftarrow \tau q_{182} \\
q_{48} &\Leftarrow \tau q_{183} \\
q_{49} &\Leftarrow \tau q_{184} \\
q_{50} &\Leftarrow \tau_c q_{98} + \tau q_{185} \\
q_{51} &\Leftarrow \tau q_{186} \\
q_{52} &\Leftarrow \tau q_{187} \\
q_{53} &\Leftarrow \tau q_{188} \\
q_{54} &\Leftarrow \tau q_{189} \\
q_{55} &\Leftarrow \tau q_{190} \\
q_{56} &\Leftarrow \tau q_{191} \\
q_{57} &\Leftarrow \tau q_{192} \\
q_{58} &\Leftarrow \tau q_{193} \\
q_{59} &\Leftarrow \tau q_{194} \\
q_{60} &\Leftarrow \tau_c q_{108} + \tau q_{195} \\
q_{61} &\Leftarrow \tau q_{196} \\
q_{62} &\Leftarrow \tau q_{197} \\
q_{63} &\Leftarrow \tau q_{198} \\
q_{64} &\Leftarrow \tau q_{199} \\
q_{65} &\Leftarrow \tau q_{200} \\
q_{66} &\Leftarrow \tau q_{201} \\
q_{67} &\Leftarrow \tau q_{202} \\
q_{68} &\Leftarrow \tau q_{203} \\
q_{69} &\Leftarrow \tau q_{204} \\
q_{70} &\Leftarrow \tau_c q_{118} + \tau q_{205} \\
q_{71} &\Leftarrow \tau q_{206} \\
q_{72} &\Leftarrow \tau q_{207} \\
q_{73} &\Leftarrow \tau q_{208} \\
q_{74} &\Leftarrow \tau q_{209} \\
q_{75} &\Leftarrow \tau q_{210} \\
q_{76} &\Leftarrow \tau q_{211} \\
q_{77} &\Leftarrow \tau q_{212} \\
q_{78} &\Leftarrow \tau q_{213} \\
q_{79} &\Leftarrow \tau q_{214} \\
q_{80} &\Leftarrow \tau_c q_{128} + \tau q_{215} \\
q_{81} &\Leftarrow \tau q_{216} \\
q_{82} &\Leftarrow \tau q_{217} \\
q_{83} &\Leftarrow \tau q_{218} \\
q_{84} &\Leftarrow \tau q_{219} \\
q_{85} &\Leftarrow \tau q_{220} \\
q_{86} &\Leftarrow \tau_c q_{134} + \tau q_{221} \\
q_{87} &\Leftarrow \tau q_{222} \\
q_{88} &\Leftarrow \tau q_{223} \\
q_{89} &\Leftarrow \tau q_{224} \\
q_{90} &\Leftarrow \tau q_{225} \\
q_{141} &\Leftarrow \tau_c q_{320} \\
q_{151} &\Leftarrow \tau_c q_{330} \\
q_{161} &\Leftarrow \tau_c q_{340} \\
q_{171} &\Leftarrow \tau_c q_{350} \\
q_{177} &\Leftarrow \tau_c q_{356} \\
q_{181} &\Leftarrow \tau_c q_{46} \\
q_{182} &\Leftarrow \tau q_{47} \\
q_{183} &\Leftarrow \tau q_{48} \\
q_{184} &\Leftarrow \tau q_{49}
\end{aligned}
$$

$$q_{185} \Leftarrow \tau_c q_{362} + \tau_c q_{233} + \tau q_{50}$$
$$q_{186} \Leftarrow \tau_c q_5 + \tau q_{51}$$
$$q_{187} \Leftarrow \tau q_{52}$$
$$q_{188} \Leftarrow \tau q_{53}$$
$$q_{189} \Leftarrow \tau q_{54}$$
$$q_{190} \Leftarrow \tau q_{55}$$
$$q_{191} \Leftarrow \tau q_{56}$$
$$q_{192} \Leftarrow \tau q_{57}$$
$$q_{193} \Leftarrow \tau q_{58}$$
$$q_{194} \Leftarrow \tau q_{59}$$
$$q_{195} \Leftarrow \tau_c q_{372} + \tau_c q_{243} + \tau q_{60}$$
$$q_{196} \Leftarrow \tau_c q_{15} + \tau q_{61}$$
$$q_{197} \Leftarrow \tau q_{62}$$
$$q_{198} \Leftarrow \tau q_{63}$$
$$q_{199} \Leftarrow \tau q_{64}$$
$$q_{200} \Leftarrow \tau q_{65}$$
$$q_{201} \Leftarrow \tau q_{66}$$
$$q_{202} \Leftarrow \tau q_{67}$$
$$q_{203} \Leftarrow \tau q_{68}$$
$$q_{204} \Leftarrow \tau q_{69}$$
$$q_{205} \Leftarrow \tau_c q_{381} + \tau_c q_{253} + \tau q_{70}$$
$$q_{206} \Leftarrow \tau_c q_{25} + \tau q_{71}$$
$$q_{207} \Leftarrow \tau q_{72}$$
$$q_{208} \Leftarrow \tau q_{73}$$
$$q_{209} \Leftarrow \tau q_{74}$$
$$q_{210} \Leftarrow \tau q_{75}$$
$$q_{211} \Leftarrow \tau q_{76}$$
$$q_{212} \Leftarrow \tau q_{77}$$
$$q_{213} \Leftarrow \tau q_{78}$$
$$q_{214} \Leftarrow \tau q_{79}$$
$$q_{215} \Leftarrow \tau_c q_{391} + \tau_c q_{263} + \tau q_{80}$$
$$q_{216} \Leftarrow \tau_c q_{35} + \tau q_{81}$$
$$q_{217} \Leftarrow \tau q_{82}$$
$$q_{218} \Leftarrow \tau q_{83}$$
$$q_{219} \Leftarrow \tau q_{84}$$
$$q_{220} \Leftarrow \tau q_{85}$$
$$q_{221} \Leftarrow \tau_c q_{382} + \tau_c q_{269} + \tau q_{86}$$
$$q_{222} \Leftarrow \tau_c q_{41} + \tau q_{87}$$
$$q_{223} \Leftarrow \tau q_{98}$$
$$q_{224} \Leftarrow \tau q_{99}$$
$$q_{225} \Leftarrow \tau q_{90}$$
$$q_{227} \Leftarrow \tau_c q_{275}$$
$$q_{229} \Leftarrow \tau_c q_{275}$$
$$q_{230} \Leftarrow \tau_c q_{92} + \tau_c q_{274} + \tau_c q_{276}$$
$$q_{231} \Leftarrow \tau_c q_{185}$$
$$q_{233} \Leftarrow \tau_c q_{275}$$
$$q_{237} \Leftarrow \tau_c q_{285}$$
$$q_{239} \Leftarrow \tau_c q_{285}$$
$$q_{240} \Leftarrow \tau_c q_{102} + \tau_c q_{284} + \tau_c q_{286}$$
$$q_{241} \Leftarrow \tau_c q_{195}$$
$$q_{243} \Leftarrow \tau_c q_{285}$$
$$q_{246} \Leftarrow \tau_c q_{295}$$
$$q_{247} \Leftarrow \tau_c q_{311}$$
$$q_{249} \Leftarrow \tau_c q_{295}$$
$$q_{250} \Leftarrow \tau_c q_{111} + \tau_c q_{294} + \tau_c q_{296}$$
$$q_{251} \Leftarrow \tau_c q_{205}$$
$$q_{253} \Leftarrow \tau_c q_{295}$$
$$q_{256} \Leftarrow \tau_c q_{305}$$
$$q_{259} \Leftarrow \tau_c q_{305}$$
$$q_{260} \Leftarrow \tau_c q_{121} + \tau_c q_{304} + \tau_c q_{306}$$
$$q_{261} \Leftarrow \tau_c q_{215}$$
$$q_{263} \Leftarrow \tau_c q_{305}$$
$$q_{265} \Leftarrow \tau_c q_{311}$$
$$q_{266} \Leftarrow \tau_c q_{112} + \tau_c q_{310} + \tau_c q_{312}$$
$$q_{267} \Leftarrow \tau_c q_{221}$$
$$q_{269} \Leftarrow \tau_c q_{311}$$
$$q_{271} \Leftarrow \tau q_{406}$$
$$q_{272} \Leftarrow \tau q_{407}$$
$$q_{273} \Leftarrow \tau q_{408}$$
$$q_{274} \Leftarrow \tau q_{409}$$
$$q_{275} \Leftarrow \tau_c q_{503} + \tau q_{410}$$
$$q_{276} \Leftarrow \tau q_{411}$$
$$q_{277} \Leftarrow \tau q_{412}$$
$$q_{278} \Leftarrow \tau q_{413}$$
$$q_{279} \Leftarrow \tau q_{414}$$
$$q_{280} \Leftarrow \tau q_{415}$$

$$q_{281} \Leftarrow \tau q_{416}$$
$$q_{282} \Leftarrow \tau q_{417}$$
$$q_{283} \Leftarrow \tau q_{418}$$
$$q_{284} \Leftarrow \tau q_{419}$$
$$q_{285} \Leftarrow \tau_c q_{513} + \tau q_{420}$$
$$q_{286} \Leftarrow \tau q_{421}$$
$$q_{287} \Leftarrow \tau q_{422}$$
$$q_{288} \Leftarrow \tau q_{423}$$
$$q_{289} \Leftarrow \tau q_{424}$$
$$q_{290} \Leftarrow \tau q_{425}$$
$$q_{291} \Leftarrow \tau q_{426}$$
$$q_{292} \Leftarrow \tau q_{427}$$
$$q_{293} \Leftarrow \tau q_{428}$$
$$q_{294} \Leftarrow \tau q_{429}$$
$$q_{295} \Leftarrow \tau_c q_{523} + \tau q_{430}$$
$$q_{296} \Leftarrow \tau q_{431}$$
$$q_{297} \Leftarrow \tau q_{432}$$
$$q_{298} \Leftarrow \tau q_{433}$$
$$q_{299} \Leftarrow \tau q_{434}$$
$$q_{300} \Leftarrow \tau q_{435}$$
$$q_{301} \Leftarrow \tau q_{436}$$
$$q_{302} \Leftarrow \tau q_{437}$$
$$q_{303} \Leftarrow \tau q_{438}$$
$$q_{304} \Leftarrow \tau q_{439}$$
$$q_{305} \Leftarrow \tau_c q_{533} + \tau q_{440}$$
$$q_{306} \Leftarrow \tau q_{441}$$
$$q_{307} \Leftarrow \tau q_{442}$$
$$q_{308} \Leftarrow \tau q_{443}$$
$$q_{309} \Leftarrow \tau q_{444}$$
$$q_{310} \Leftarrow \tau q_{445}$$
$$q_{311} \Leftarrow \tau_c q_{539} + \tau q_{446}$$
$$q_{312} \Leftarrow \tau q_{447}$$
$$q_{313} \Leftarrow \tau q_{448}$$
$$q_{314} \Leftarrow \tau q_{449}$$
$$q_{315} \Leftarrow \tau q_{450}$$
$$q_{317} \Leftarrow \tau_c q_{320}$$
$$q_{319} \Leftarrow \tau_c q_{320}$$
$$q_{320} \Leftarrow \tau_c q_{497} + \tau_c q_{319} + \tau_c q_{321} + \tau_c q_{143}$$
$$q_{323} \Leftarrow \tau_c q_{320}$$
$$q_{327} \Leftarrow \tau_c q_{330}$$
$$q_{329} \Leftarrow \tau_c q_{330}$$
$$q_{330} \Leftarrow \tau_c q_{507} + \tau_c q_{329} + \tau_c q_{331} + \tau_c q_{153}$$
$$q_{333} \Leftarrow \tau_c q_{330}$$
$$q_{336} \Leftarrow \tau_c q_{340}$$
$$q_{337} \Leftarrow \tau_c q_{356}$$
$$q_{339} \Leftarrow \tau_c q_{340}$$
$$q_{340} \Leftarrow \tau_c q_{516} + \tau_c q_{339} + \tau_c q_{341} + \tau_c q_{163}$$
$$q_{343} \Leftarrow \tau_c q_{340}$$
$$q_{346} \Leftarrow \tau_c q_{350}$$
$$q_{349} \Leftarrow \tau_c q_{350}$$
$$q_{350} \Leftarrow \tau_c q_{526} + \tau_c q_{349} + \tau_c q_{351} + \tau_c q_{173}$$
$$q_{353} \Leftarrow \tau_c q_{350}$$
$$q_{355} \Leftarrow \tau_c q_{356}$$
$$q_{356} \Leftarrow \tau_c q_{517} + \tau_c q_{355} + \tau_c q_{357} + \tau_c q_{179}$$
$$q_{359} \Leftarrow \tau_c q_{356}$$
$$q_{365} \Leftarrow \tau_c q_{227}$$
$$q_{375} \Leftarrow \tau_c q_{237}$$
$$q_{385} \Leftarrow \tau_c q_{246}$$
$$q_{395} \Leftarrow \tau_c q_{256}$$
$$q_{401} \Leftarrow \tau_c q_{247}$$
$$q_{406} \Leftarrow \tau q_{271}$$
$$q_{407} \Leftarrow \tau q_{272}$$
$$q_{408} \Leftarrow \tau q_{273}$$
$$q_{409} \Leftarrow \tau q_{274}$$
$$q_{410} \Leftarrow \tau_c q_{452} + \tau q_{275}$$
$$q_{411} \Leftarrow \tau q_{276}$$
$$q_{412} \Leftarrow \tau q_{277}$$
$$q_{413} \Leftarrow \tau q_{278}$$
$$q_{414} \Leftarrow \tau q_{279}$$
$$q_{415} \Leftarrow \tau q_{280}$$
$$q_{416} \Leftarrow \tau q_{281}$$
$$q_{417} \Leftarrow \tau q_{282}$$
$$q_{418} \Leftarrow \tau q_{283}$$
$$q_{419} \Leftarrow \tau q_{284}$$
$$q_{420} \Leftarrow \tau_c q_{462} + \tau q_{285}$$
$$q_{421} \Leftarrow \tau q_{286}$$

$$q_{422} \Leftarrow \tau q_{287}$$
$$q_{423} \Leftarrow \tau q_{288}$$
$$q_{424} \Leftarrow \tau q_{289}$$
$$q_{425} \Leftarrow \tau q_{290}$$
$$q_{426} \Leftarrow \tau q_{291}$$
$$q_{427} \Leftarrow \tau q_{292}$$
$$q_{428} \Leftarrow \tau q_{293}$$
$$q_{429} \Leftarrow \tau q_{294}$$
$$q_{430} \Leftarrow \tau_c q_{471} + \tau q_{295}$$
$$q_{431} \Leftarrow \tau q_{296}$$
$$q_{432} \Leftarrow \tau q_{297}$$
$$q_{433} \Leftarrow \tau q_{298}$$
$$q_{434} \Leftarrow \tau q_{299}$$
$$q_{435} \Leftarrow \tau q_{300}$$
$$q_{436} \Leftarrow \tau q_{301}$$
$$q_{437} \Leftarrow \tau q_{302}$$
$$q_{438} \Leftarrow \tau q_{303}$$
$$q_{439} \Leftarrow \tau q_{304}$$
$$q_{440} \Leftarrow \tau_c q_{481} + \tau q_{305}$$
$$q_{441} \Leftarrow \tau q_{306}$$
$$q_{442} \Leftarrow \tau q_{307}$$
$$q_{443} \Leftarrow \tau q_{308}$$
$$q_{444} \Leftarrow \tau q_{309}$$
$$q_{445} \Leftarrow \tau q_{310}$$
$$q_{446} \Leftarrow \tau_c q_{472} + \tau q_{311}$$
$$q_{447} \Leftarrow \tau q_{312}$$
$$q_{448} \Leftarrow \tau q_{313}$$
$$q_{449} \Leftarrow \tau q_{314}$$
$$q_{450} \Leftarrow \tau q_{315}$$
$$q_{501} \Leftarrow \tau_c q_{275}$$
$$q_{511} \Leftarrow \tau_c q_{285}$$
$$q_{521} \Leftarrow \tau_c q_{295}$$
$$q_{531} \Leftarrow \tau_c q_{305}$$
$$q_{537} \Leftarrow \tau_c q_{311}$$

The τ_i-actions are identified using the symmetry rules. The resulting structure is now ready for the splitting process. The τ-splitting process gives :-

$$
\begin{aligned}
q_2 &\Leftarrow < p_{11} \rightarrow^{\alpha} p_{14} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > \\
q_4 &\Leftarrow < p_{13} \rightarrow^{\alpha} p_{14} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > \\
q_5 &\Leftarrow < p_{14} \rightarrow^{\alpha} p_{13} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > + < p_{14} \rightarrow^{\alpha} p_{15} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > + < p_{14} \rightarrow^{\mu} p_{17} \mid r_1 \rightarrow^{\overline{\mu}} r_5 > \\
q_8 &\Leftarrow < p_{17} \rightarrow^{\alpha} p_{14} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > \\
q_{12} &\Leftarrow < p_{20} \rightarrow^{\alpha} p_{23} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > \\
q_{14} &\Leftarrow < p_{22} \rightarrow^{\alpha} p_{23} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > \\
q_{15} &\Leftarrow < p_{23} \rightarrow^{\alpha} p_{22} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > + < p_{23} \rightarrow^{\alpha} p_{24} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > + < p_{23} \rightarrow^{\mu} p_{26} \mid r_1 \rightarrow^{\overline{\mu}} r_5 > \\
q_{18} &\Leftarrow < p_{26} \rightarrow^{\alpha} p_{23} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > \\
q_{21} &\Leftarrow < p_{29} \rightarrow^{\alpha} p_{32} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > \\
q_{22} &\Leftarrow < p_2 \rightarrow^{\alpha} p_5 \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > \\
q_{24} &\Leftarrow < p_{31} \rightarrow^{\alpha} p_{32} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > \\
q_{25} &\Leftarrow < p_{32} \rightarrow^{\alpha} p_{31} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > + < p_{32} \rightarrow^{\alpha} p_{33} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > + < p_{32} \rightarrow^{\mu} p_{35} \mid r_1 \rightarrow^{\overline{\mu}} r_5 > \\
q_{28} &\Leftarrow < p_{35} \rightarrow^{\alpha} p_{32} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > \\
q_{31} &\Leftarrow < p_{38} \rightarrow^{\alpha} p_{41} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > \\
q_{34} &\Leftarrow < p_{40} \rightarrow^{\alpha} p_{41} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > \\
q_{35} &\Leftarrow < p_{41} \rightarrow^{\alpha} p_{40} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > + < p_{41} \rightarrow^{\alpha} p_{42} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > + < p_{41} \rightarrow^{\mu} p_{44} \mid r_1 \rightarrow^{\overline{\mu}} r_5 > \\
q_{38} &\Leftarrow < p_{44} \rightarrow^{\alpha} p_{41} \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > \\
q_{40} &\Leftarrow < p_4 \rightarrow^{\alpha} p_5 \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > \\
q_{41} &\Leftarrow < p_5 \rightarrow^{\alpha} p_4 \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > + < p_5 \rightarrow^{\alpha} p_6 \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > + < p_5 \rightarrow^{\mu} p_8 \mid r_1 \rightarrow^{\overline{\mu}} r_5 > \\
q_{44} &\Leftarrow < p_8 \rightarrow^{\alpha} p_5 \mid r_1 \rightarrow^{\overline{\alpha}} r_2 > \\
q_{50} &\Leftarrow < p_{14} \rightarrow^{\mu} p_{17} \mid r_2 \rightarrow^{\overline{\mu}} r_3 > \\
q_{60} &\Leftarrow < p_{23} \rightarrow^{\mu} p_{26} \mid r_2 \rightarrow^{\overline{\mu}} r_3 > \\
q_{70} &\Leftarrow < p_{32} \rightarrow^{\mu} p_{35} \mid r_2 \rightarrow^{\overline{\mu}} r_3 > \\
q_{80} &\Leftarrow < p_{41} \rightarrow^{\mu} p_{44} \mid r_2 \rightarrow^{\overline{\mu}} r_3 > \\
q_{86} &\Leftarrow < p_5 \rightarrow^{\mu} p_8 \mid r_2 \rightarrow^{\overline{\mu}} r_3 > \\
q_{141} &\Leftarrow < p_{15} \rightarrow^{\theta} p_{14} \mid r_4 \rightarrow^{\overline{\theta}} r_8 > \\
q_{151} &\Leftarrow < p_{24} \rightarrow^{\theta} p_{23} \mid r_4 \rightarrow^{\overline{\theta}} r_8 > \\
q_{161} &\Leftarrow < p_{33} \rightarrow^{\theta} p_{32} \mid r_4 \rightarrow^{\overline{\theta}} r_8 > \\
q_{171} &\Leftarrow < p_{42} \rightarrow^{\theta} p_{41} \mid r_4 \rightarrow^{\overline{\theta}} r_8 > \\
q_{177} &\Leftarrow < p_6 \rightarrow^{\theta} p_5 \mid r_4 \rightarrow^{\overline{\theta}} r_8 > \\
q_{185} &\Leftarrow < p_{14} \rightarrow^{\delta} p_{11} \mid r_5 \rightarrow^{\overline{\delta}} r_9 > + < p_{14} \rightarrow^{\mu} p_{17} \mid r_5 \rightarrow^{\overline{\mu}} r_6 > \\
q_{186} &\Leftarrow < p_{15} \rightarrow^{\theta} p_{14} \mid r_5 \rightarrow^{\overline{\theta}} r_1 > \\
q_{195} &\Leftarrow < p_{23} \rightarrow^{\delta} p_{20} \mid r_5 \rightarrow^{\overline{\delta}} r_9 > + < p_{23} \rightarrow^{\mu} p_{26} \mid r_5 \rightarrow^{\overline{\mu}} r_6 > \\
q_{196} &\Leftarrow < p_{24} \rightarrow^{\theta} p_{23} \mid r_5 \rightarrow^{\overline{\theta}} r_1 >
\end{aligned}
$$

$$q_{205} \Leftarrow < p_{32} \rightarrow^{\delta} p_{29} \mid r_5 \rightarrow^{\overline{\delta}} r_9 > + < p_{32} \rightarrow^{\mu} p_{35} \mid r_5 \rightarrow^{\overline{\mu}} r_6 >$$

$$q_{206} \Leftarrow < p_{33} \rightarrow^{\theta} p_{32} \mid r_5 \rightarrow^{\overline{\theta}} r_1 >$$

$$q_{215} \Leftarrow < p_{41} \rightarrow^{\delta} p_{38} \mid r_5 \rightarrow^{\overline{\delta}} r_9 > + < p_{41} \rightarrow^{\mu} p_{44} \mid r_5 \rightarrow^{\overline{\mu}} r_6 >$$

$$q_{216} \Leftarrow < p_{42} \rightarrow^{\theta} p_{41} \mid r_5 \rightarrow^{\overline{\theta}} r_1 >$$

$$q_{221} \Leftarrow < p_5 \rightarrow^{\delta} p_2 \mid r_5 \rightarrow^{\overline{\delta}} r_9 > + < p_5 \rightarrow^{\mu} p_8 \mid r_5 \rightarrow^{\overline{\mu}} r_6 >$$

$$q_{222} \Leftarrow < p_6 \rightarrow^{\theta} p_5 \mid r_5 \rightarrow^{\overline{\theta}} r_1 >$$

$$q_{227} \Leftarrow < p_{11} \rightarrow^{\alpha} p_{14} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{229} \Leftarrow < p_{13} \rightarrow^{\alpha} p_{14} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{230} \Leftarrow < p_{14} \rightarrow^{\delta} p_{11} \mid r_6 \rightarrow^{\overline{\delta}} r_3 > + < p_{14} \rightarrow^{\alpha} p_{13} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 > + < p_{14} \rightarrow^{\alpha} p_{15} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{231} \Leftarrow < p_{15} \rightarrow^{\theta} p_{14} \mid r_6 \rightarrow^{\overline{\theta}} r_5 >$$

$$q_{233} \Leftarrow < p_{17} \rightarrow^{\alpha} p_{14} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{237} \Leftarrow < p_{20} \rightarrow^{\alpha} p_{23} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{239} \Leftarrow < p_{22} \rightarrow^{\alpha} p_{23} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{240} \Leftarrow < p_{23} \rightarrow^{\delta} p_{20} \mid r_6 \rightarrow^{\overline{\delta}} r_3 > + < p_{23} \rightarrow^{\alpha} p_{22} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 > + < p_{23} \rightarrow^{\alpha} p_{24} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{241} \Leftarrow < p_{24} \rightarrow^{\theta} p_{23} \mid r_6 \rightarrow^{\overline{\theta}} r_5 >$$

$$q_{243} \Leftarrow < p_{26} \rightarrow^{\alpha} p_{23} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{246} \Leftarrow < p_{29} \rightarrow^{\alpha} p_{32} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{247} \Leftarrow < p_2 \rightarrow^{\alpha} p_5 \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{249} \Leftarrow < p_{31} \rightarrow^{\alpha} p_{32} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{250} \Leftarrow < p_{32} \rightarrow^{\delta} p_{29} \mid r_6 \rightarrow^{\overline{\delta}} r_3 > + < p_{32} \rightarrow^{\alpha} p_{31} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 > + < p_{32} \rightarrow^{\alpha} p_{33} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{251} \Leftarrow < p_{33} \rightarrow^{\theta} p_{32} \mid r_6 \rightarrow^{\overline{\theta}} r_5 >$$

$$q_{253} \Leftarrow < p_{35} \rightarrow^{\alpha} p_{32} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{256} \Leftarrow < p_{38} \rightarrow^{\alpha} p_{41} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{259} \Leftarrow < p_{40} \rightarrow^{\alpha} p_{41} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{260} \Leftarrow < p_{41} \rightarrow^{\delta} p_{38} \mid r_6 \rightarrow^{\overline{\delta}} r_3 > + < p_{41} \rightarrow^{\alpha} p_{40} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 > + < p_{41} \rightarrow^{\alpha} p_{42} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{261} \Leftarrow < p_{42} \rightarrow^{\theta} p_{41} \mid r_6 \rightarrow^{\overline{\theta}} r_5 >$$

$$q_{263} \Leftarrow < p_{44} \rightarrow^{\alpha} p_{41} \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{265} \Leftarrow < p_4 \rightarrow^{\alpha} p_5 \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{266} \Leftarrow < p_5 \rightarrow^{\delta} p_2 \mid r_6 \rightarrow^{\overline{\delta}} r_3 > + < p_5 \rightarrow^{\alpha} p_4 \mid r_6 \rightarrow^{\overline{\alpha}} r_7 > + < p_5 \rightarrow^{\alpha} p_6 \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{267} \Leftarrow < p_6 \rightarrow^{\theta} p_5 \mid r_6 \rightarrow^{\overline{\theta}} r_5 >$$

$$q_{269} \Leftarrow < p_8 \rightarrow^{\alpha} p_5 \mid r_6 \rightarrow^{\overline{\alpha}} r_7 >$$

$$q_{275} \Leftarrow < p_{14} \rightarrow^{\mu} p_{17} \mid r_7 \rightarrow^{\overline{\mu}} r_{12} >$$

$$q_{285} \Leftarrow < p_{23} \rightarrow^{\mu} p_{26} \mid r_7 \rightarrow^{\overline{\mu}} r_{12} >$$

$$q_{295} \Leftarrow < p_{32} \rightarrow^{\mu} p_{35} \mid r_7 \rightarrow^{\overline{\mu}} r_{12} >$$

$$q_{305} \Leftarrow < p_{41} \rightarrow^{\mu} p_{44} \mid r_7 \rightarrow^{\overline{\mu}} r_{12} >$$

$$q_{311} \Leftarrow < p_5 \rightarrow^{\mu} p_8 \mid r_7 \rightarrow^{\overline{\mu}} r_{12} >$$

$$q_{317} \Leftarrow < p_{11} \rightarrow^{\alpha} p_{14} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 >$$

$$q_{319} \Leftarrow < p_{13} \rightarrow^{\alpha} p_{14} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 >$$

$$q_{320} \Leftarrow < p_{14} \rightarrow^{\delta} p_{11} \mid r_8 \rightarrow^{\overline{\delta}} r_{12} > + < p_{14} \rightarrow^{\alpha} p_{13} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 > + < p_{14} \rightarrow^{\alpha} p_{15} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 > + < p_{14} \rightarrow^{\mu} p_{17} \mid r_8 \rightarrow^{\overline{\mu}} r_4$$

$$q_{323} \Leftarrow < p_{17} \rightarrow^{\alpha} p_{14} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 >$$

$$q_{327} \Leftarrow < p_{20} \rightarrow^{\alpha} p_{23} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 >$$

$$q_{329} \Leftarrow < p_{22} \rightarrow^{\alpha} p_{23} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 >$$

$$q_{330} \Leftarrow < p_{23} \rightarrow^{\delta} p_{20} \mid r_8 \rightarrow^{\overline{\delta}} r_{12} > + < p_{23} \rightarrow^{\alpha} p_{22} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 > + < p_{23} \rightarrow^{\alpha} p_{24} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 > + < p_{23} \rightarrow^{\mu} p_{26} \mid r_8 \rightarrow^{\overline{\mu}} r_4$$

$$q_{333} \Leftarrow < p_{26} \rightarrow^{\alpha} p_{23} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 >$$

$$q_{336} \Leftarrow < p_{29} \rightarrow^{\alpha} p_{32} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 >$$

$$q_{337} \Leftarrow < p_2 \rightarrow^{\alpha} p_5 \mid r_8 \rightarrow^{\overline{\alpha}} r_8 >$$

$$q_{339} \Leftarrow < p_{31} \rightarrow^{\alpha} p_{32} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 >$$

$$q_{340} \Leftarrow < p_{32} \rightarrow^{\delta} p_{29} \mid r_8 \rightarrow^{\overline{\delta}} r_{12} > + < p_{32} \rightarrow^{\alpha} p_{31} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 > + < p_{32} \rightarrow^{\alpha} p_{33} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 > + < p_{32} \rightarrow^{\mu} p_{35} \mid r_8 \rightarrow^{\overline{\mu}} r_4$$

$$q_{343} \Leftarrow < p_{35} \rightarrow^{\alpha} p_{32} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 >$$

$$q_{346} \Leftarrow < p_{38} \rightarrow^{\alpha} p_{41} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 >$$

$$q_{349} \Leftarrow < p_{40} \rightarrow^{\alpha} p_{41} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 >$$

$$q_{350} \Leftarrow < p_{41} \rightarrow^{\delta} p_{38} \mid r_8 \rightarrow^{\overline{\delta}} r_{12} > + < p_{41} \rightarrow^{\alpha} p_{40} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 > + < p_{41} \rightarrow^{\alpha} p_{42} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 > + < p_{41} \rightarrow^{\mu} p_{44} \mid r_8 \rightarrow^{\overline{\mu}} r_4$$

$$q_{353} \Leftarrow < p_{44} \rightarrow^{\alpha} p_{41} \mid r_8 \rightarrow^{\overline{\alpha}} r_8 >$$

$$q_{355} \Leftarrow < p_4 \rightarrow^{\alpha} p_5 \mid r_8 \rightarrow^{\overline{\alpha}} r_8 >$$

$$q_{356} \Leftarrow < p_5 \rightarrow^{\delta} p_{12} \mid r_8 \rightarrow^{\overline{\delta}} r_{12} > + < p_5 \rightarrow^{\alpha} p_4 \mid r_8 \rightarrow^{\overline{\alpha}} r_8 > + < p_5 \rightarrow^{\alpha} p_6 \mid r_8 \rightarrow^{\overline{\alpha}} r_8 > + < p_5 \rightarrow^{\mu} p_8 \mid r_8 \rightarrow^{\overline{\mu}} r_4$$

$$q_{359} \Leftarrow < p_8 \rightarrow^{\alpha} p_5 \mid r_8 \rightarrow^{\overline{\alpha}} r_8 >$$

$$q_{365} \Leftarrow < p_{14} \rightarrow^{\delta} p_{11} \mid r_9 \rightarrow^{\overline{\delta}} r_6 >$$

$$q_{375} \Leftarrow < p_{23} \rightarrow^{\delta} p_{20} \mid r_9 \rightarrow^{\overline{\delta}} r_6 >$$

$$q_{385} \Leftarrow < p_{32} \rightarrow^{\delta} p_{29} \mid r_9 \rightarrow^{\overline{\delta}} r_6 >$$

$$q_{395} \Leftarrow < p_{41} \rightarrow^{\delta} p_{38} \mid r_9 \rightarrow^{\overline{\delta}} r_6 >$$

$$q_{401} \Leftarrow < p_5 \rightarrow^{\delta} p_2 \mid r_9 \rightarrow^{\overline{\delta}} r_6 >$$

$$q_{410} \Leftarrow < p_{14} \rightarrow^{\delta} p_{11} \mid r_{10} \rightarrow^{\overline{\delta}} r_{11} >$$

$$q_{420} \Leftarrow < p_{23} \rightarrow^{\delta} p_{20} \mid r_{10} \rightarrow^{\overline{\delta}} r_{11} >$$

$$q_{430} \Leftarrow < p_{32} \rightarrow^{\delta} p_{29} \mid r_{10} \rightarrow^{\overline{\delta}} r_{11} >$$

$$q_{440} \Leftarrow < p_{41} \rightarrow^{\delta} p_{38} \mid r_{10} \rightarrow^{\overline{\delta}} r_{11} >$$

$$q_{446} \Leftarrow < p_5 \rightarrow^{\delta} p_2 \mid r_{10} \rightarrow^{\overline{\delta}} r_{11} >$$

$$q_{501} \Leftarrow < p_{15} \rightarrow^{\theta} p_{14} \mid r_{12} \rightarrow^{\overline{\theta}} r_7 >$$

$$q_{511} \Leftarrow < p_{24} \rightarrow^{\theta} p_{23} \mid r_{12} \rightarrow^{\overline{\theta}} r_7 >$$
$$q_{521} \Leftarrow < p_{33} \rightarrow^{\theta} p_{32} \mid r_{12} \rightarrow^{\overline{\theta}} r_7 >$$
$$q_{531} \Leftarrow < p_{42} \rightarrow^{\theta} p_{41} \mid r_{12} \rightarrow^{\overline{\theta}} r_7 >$$
$$q_{537} \Leftarrow < p_{6} \rightarrow^{\theta} p_{5} \mid r_{12} \rightarrow^{\overline{\theta}} r_7 >$$

The other non-τ actions of $\mathcal{M}_q$ can now be re-inserted and the adjacency matrices of $\mathcal{M}_p$ and $\mathcal{M}_q$ derived. Due to the size of the $\mathcal{M}_q$ machine, this will be done for $\mathcal{M}_p$ only. The principle is exactly the same for $\mathcal{M}_q$.

Referring back to the specification for $\mathcal{M}_p$, we need group multiple actions together. This occurs when there are multiple paths between events. For this example this does not occur. In addition, for the algorithms considered here, only unlabelled digraphs can be used. Hence all the actions can be removed (once the grouping has been carried out), which gives :-

$$p_1 \Leftarrow p_1 + p_4 + p_{10}$$
$$p_2 \Leftarrow p_1 + p_5$$
$$p_3 \Leftarrow p_2 + p_3 + p_{19}$$
$$p_4 \Leftarrow p_5 + p_7$$
$$p_5 \Leftarrow p_4 + p_6 + p_2 + p_8$$
$$p_6 \Leftarrow p_5 + p_3$$
$$p_7 \Leftarrow p_7 + p_8 + p_{28}$$
$$p_8 \Leftarrow p_5 + p_9$$
$$p_9 \Leftarrow p_6 + p_9 + p_{37}$$
$$p_{10} \Leftarrow p_{13} + p_1$$
$$p_{11} \Leftarrow p_{10} + p_{14}$$
$$p_{12} \Leftarrow p_{11}$$
$$p_{13} \Leftarrow p_{14} + p_{16}$$
$$p_{14} \Leftarrow p_{13} + p_{15} + p_{11} + p_{17}$$
$$p_{15} \Leftarrow p_{14} + p_{12}$$
$$p_{16} \Leftarrow p_{17}$$
$$p_{17} \Leftarrow p_{14} + p_{18}$$
$$p_{18} \Leftarrow p_{15}$$
$$p_{19} \Leftarrow p_{22} + p_3$$
$$p_{20} \Leftarrow p_{19} + p_{23}$$
$$p_{21} \Leftarrow p_{20}$$
$$p_{22} \Leftarrow p_{23} + p_{25}$$
$$p_{23} \Leftarrow p_{22} + p_{24} + p_{20} + p_{26}$$
$$p_{24} \Leftarrow p_{23} + p_{21}$$
$$p_{25} \Leftarrow p_{26}$$
$$p_{26} \Leftarrow p_{23} + p_{27}$$

$$
\begin{aligned}
p_{27} &\Leftarrow p_{24} \\
p_{28} &\Leftarrow p_{31} + p_7 \\
p_{29} &\Leftarrow p_{28} + p_{32} \\
p_{30} &\Leftarrow p_{29} \\
p_{31} &\Leftarrow p_{32} + p_{34} \\
p_{32} &\Leftarrow p_{31} + p_{33} + p_{29} + p_{35} \\
p_{33} &\Leftarrow p_{32} + p_{30} \\
p_{34} &\Leftarrow p_{35} \\
p_{35} &\Leftarrow p_{32} + p_{36} \\
p_{36} &\Leftarrow p_{33} \\
p_{37} &\Leftarrow p_{40} + p_9 \\
p_{38} &\Leftarrow p_{37} + p_{41} \\
p_{39} &\Leftarrow p_{38} \\
p_{40} &\Leftarrow p_{41} + p_{43} \\
p_{41} &\Leftarrow p_{40} + p_{42} + p_{38} + p_{44} \\
p_{42} &\Leftarrow p_{41} + p_{39} \\
p_{43} &\Leftarrow p_{44} \\
p_{44} &\Leftarrow p_{41} + p_{45} \\
p_{45} &\Leftarrow p_{42}
\end{aligned}
$$

The associated adjacency matrix is easily derived from this, being mapped directly from this system of equations.

The adjacency matrix, and that for $\mathcal{M}_q$ is input into the quotient algorithm. The output is the prime factors of $\mathcal{M}_q$ and $\mathcal{M}_p$. Factoring out from the prime factorisation sequence of $\mathcal{M}_q$ the prime factors of $\mathcal{M}_p$, leaves a residue. It is this residue that is the prime factor sequence of $\mathcal{M}_r$. There are a number of graph theory tools which will produce the cartesian product of a set of given graphs, for example daVinci [243], GAP [224] and LEDA [158].

The adjacency matrix for $\mathcal{M}_r$ is :-

$$
\begin{pmatrix}
010010000000 \\
001010000000 \\
000101000000 \\
000001020000 \\
210011001000 \\
011010101000 \\
000000000101 \\
000100010001 \\
000011000000 \\
000001100010 \\
000000000100 \\
000000210000
\end{pmatrix}
$$

The associated digraph of $\mathcal{M}_r$ is given in figure 8.2.

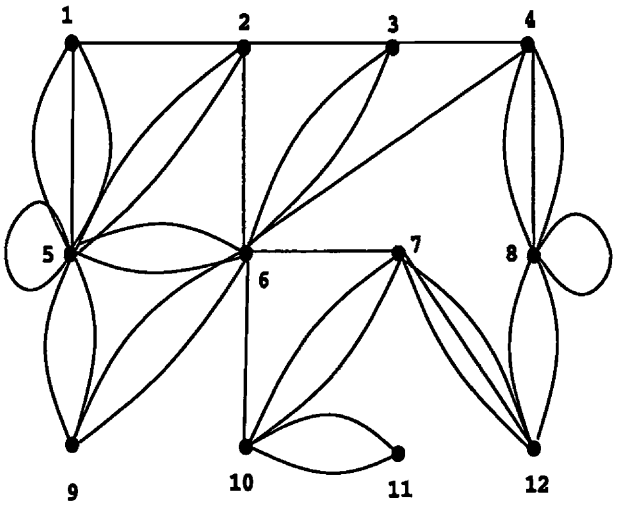


Figure 8.2: The underlying digraph structure of $\mathcal{M}_r$

After relabelling, and converting back to CCS prefix notation, we can present the complete specification of $\mathcal{M}_r$:-

$$
\begin{aligned}
r_1 &\Leftarrow \overline{\alpha} r_2 + \overline{\mu} r_5 \\
r_2 &\Leftarrow \overline{\mu} r_3 + \tau r_5 \\
r_3 &\Leftarrow \eta r_4 + \gamma r_6 \\
r_4 &\Leftarrow \overline{\theta} r_8 + \gamma r_6 + \eta r_8 \\
r_5 &\Leftarrow \beta r_5 + \eta r_1 + \overline{\theta} r_1 + \tau r_2 + \overline{\delta} r_9 + \overline{\mu} r_6 \\
r_6 &\Leftarrow \beta r_2 + \overline{\delta} r_3 + \overline{\alpha} r_7 + \gamma r_9 + \overline{\theta} r_5 \\
r_7 &\Leftarrow \tau r_{10} + \overline{\mu} r_{12} \\
r_8 &\Leftarrow \overline{\alpha} r_8 + \overline{\mu} r_4 + \overline{\delta} r_{12} \\
r_9 &\Leftarrow \gamma r_5 + \overline{\delta} r_6 \\
r_{10} &\Leftarrow \beta r_6 + \tau r_7 + \overline{\delta} r_{11} \\
r_{11} &\Leftarrow \gamma r_{10} \\
r_{12} &\Leftarrow \gamma r_8 + \eta r_7 + \overline{\theta} r_7
\end{aligned}
$$

This is the machine which will enable the protocol as represented by $\mathcal{M}_p$ to interface with that of $\mathcal{M}_q$.

C.4 Summary

In terms of past literature, most researchers have applied their theories to generally small examples. Typically, one sees the ABP, CSMA etc used. Whilst providing suitable vehicles in terms of promoting understanding, they leave open issues such as generality and scalability.

Application to a large example was considered to be a key part of this research. In fact, the hope was to apply the theory to real, industrial examples, such as DPNSS and DASS2 (at least 2000 states). Unfortunately these problems were specified in English, and the effort involved in converting them to CCS was considered prohibitive. Some protocols were available in LOTOS, but these were not neither large enough or complete. Hence, the alternative option - to produce an artificial example that nevertheless would include many problematic characteristics that would probably not be encountered in real protocols, at least *en masse*.

The conclusions to be drawn from this example is that the theory appears to be robust, but that there are a number of issues that need further study. The example took in the region of 6 hours of continuous effort, starting from the given specifications to the complete $\mathcal{M}_\tau$. There is good reason to believe that with tool support this could quite conservatively be reduced to less than 1 hour, and if fully automated we would be likely looking at minutes or even seconds. In the above example, since it is relatively straightforward, the strategy would be to use the directed, but unlabelled graph quotient algorithms and to derive the complete $\mathcal{M}_\tau$ machine using pattern matching on the original $\mathcal{M}_q^{540}$. Such algorithms have polynomial time complexity and as such would have minimal overhead in terms of the end-to-end processing. Indeed, it is likely that the pre and post processing algorithms would constitute the bulk of the overall processing time. For instance, whilst the τ-splitting algorithms would probably be quite efficient, the unfolding algorithm may have to be semi-heuristic.

D Observational equivalence

To generalise the work we do need to find a transformation χ such that :-

$$
\begin{array}{lrcll}
\text{Given} & (\mathcal{M}_p \mid \mathcal{M}_r) \wr A \backslash B & \approx & \mathcal{M}_q & \text{then } \exists \chi : \\
& \chi(\mathcal{M}_q) & = & \mathcal{M}'_q & \text{and} \\
& \chi(\mathcal{M}_p) & = & \mathcal{M}'_p & \\
\Rightarrow & (\mathcal{M}'_p \mid \mathcal{M}_r) \wr A \backslash B & \sim & \mathcal{M}'_q & \text{and via } \mathbf{Sym} : \\
& \mathbf{Sym}(\mathcal{M}'_q) & = & \mathcal{M}''_q & \text{and} \\
& \mathbf{Sym}(\mathcal{M}'_p) & = & \mathcal{M}''_p & \\
\Rightarrow & (\mathcal{M}''_p \mid \mathcal{M}_r) \wr A \backslash B & \stackrel{\mathrm{S}}{=} & \mathcal{M}''_q & \text{and via} \Upsilon : \\
& \Upsilon(\mathcal{M}''_q) & = & \mathcal{M}'''_q & \text{and} \\
& \Upsilon(\mathcal{M}''_p) & = & \mathcal{M}'''_p & \\
\Rightarrow & (\mathcal{M}'''_p \mid \mathcal{M}_r) \backslash B & \cong & \mathcal{M}'''_q & \\
\Rightarrow & \mathcal{G}_{Mp'''} \times \mathcal{G}_{\mathcal{M}_r} & \cong & \mathcal{G}_{\mathcal{M}'''_q} & \text{by Theorem 5.3.2 and Cors 5.3.4 and 5.3.3.}
\end{array}
$$

In the above equations we have two transformations χ and **Sym** which take an observationally equivalent interface equation, and transforms it via a strongly equivalent equation, to the desired structural equivalence. Once the equation has reached this stage tau splitting can be applied. The following outlines an approach which will effectively bypass the strongly equivalent state, and transform observational equivalence to structural equivalence directly. The approach is interesting not only from the point of view of the transform problem, but also because it requires the notion of a metric space. Researchers such as Francez, Rounds and Golson have produced some interesting work examining concurrent machines, process algebras and metric spaces [78, 198, 199, 82].

Assume we are given an interface equation of the form $(\mathcal{M}_p \mid \mathcal{M}_r) \wr A \backslash B \approx \mathcal{M}_q$. Since we are dealing with observational equivalence, the machines $\mathcal{M}_p$ and $\mathcal{M}_q$ can be subjected to reduction. Now consider the machine $\mathcal{M}_q - \{\tau_c\}$. As before, we first establish its sequence of $\tilde{\Pi}$-planes, $\tilde{\Pi}_1 \ldots \tilde{\Pi}_n$, which will in general not be structurally equivalent (remember, this is a requirement for the quotient approach). Let us construct the machine $\mathcal{M}^*_p = \mathcal{M}_p - A \cup \{\tau_c\}$, which provides some idea of the likely internal structure of the $\tilde{\Pi}$-planes. The idea behind the postulated approach is to find the $\tilde{\Pi}$-plane which is closest (via a metric) to $\mathcal{M}^*_p$, and to replace all other $\tilde{\Pi}$-plane with this set. This metric will provide us with a distance measure between machines, or their associated state transition graphs. The actual details of the measure should not concern us greatly here, suffice to say that we assume such a measure can be found, and that the notion of distance between machines and graphs is a meaningful one. We define the metric then, in general as :-

D.1 Definition

$$\partial(\tilde{\Pi}_i, \mathcal{M}_p^*) = \begin{cases} s & \text{if } \tilde{\Pi}_i \overset{S}{\neq} \mathcal{M}_p^* \\ 0 & \text{if } \tilde{\Pi}_i \overset{S}{=} \mathcal{M}_p^* \end{cases}$$

Let $\tilde{\Pi}_{min} = \min(\partial(\tilde{\Pi}_i, \mathcal{M}_p^*))$. Clearly $\tilde{\Pi}_i \subseteq \tilde{\Pi}_{min} \forall$ i . Repeat this procedure and construct the set L_{max}, though of course in the later case we have little idea of the structure of $\mathcal{M}_r$ - we merely construct the maximal L. Returning to the interface equation we now now replace $\mathcal{M}_p$ by $\tilde{\Pi}_{min}$ and $\mathcal{M}_r$ by L_{max}. We replace observational equivalence with structural equivalence to give :-

$$(\tilde{\Pi}_{min} \mid L_{min}^{\circ}) \wr A \backslash B \overset{S}{=} \mathcal{M}_q'$$

The communicating τ actions are then replaced, and the quotient extracted via the theory described in this thesis. The crux of this approach is that $\tilde{\Pi}_{min}$ and L_{max} are approximations to the structure of $\mathcal{M}_p$ and $\mathcal{M}_r$ respectively expected under structural equivalence. Since the $\tilde{\Pi}$-planes are now all isomorphic the quotient graph approach can be used after τ-splitting. However, the solution for $\mathcal{M}_r$ will itself be approximate. The question is whether we can establish some iterative mechanism (see 7) whereby the equation converges to a solution or diverges. The concept of a metric space is particularly important here. Our view is that we adopt an approach similar to that of Francez in [78]. In this example we have to modify the $\mathcal{M}_p$ machine in order to establish a stable solution $\mathcal{M}_r$. What we would really like to do is to make an informed guess for $\mathcal{M}_r$, and to see whether this initial approximation converges to a solution, akin to Martin [142].

E Iterative solution methods

As a starting point, consider the interface equation :-

$$(\mathcal{M}_p \mid \mathcal{M}_r) \wr A \backslash B \approx \mathcal{M}_q$$

Assume $\mathcal{M}'_r$ is our initial estimate for $\mathcal{M}_r$. We thus have :-

$$(\mathcal{M}_p \mid \mathcal{M}'_r) \wr A \backslash B \approx \mathcal{M}_q$$

Evaluating both sides, what is the difference between the LHS and RHS? Re-arrange the equation to :-

$$(\mathcal{M}_p \mid \mathcal{M}'_r) \wr A \backslash B - \mathcal{M}_q \approx \mathbf{Res}$$

This residue **Res** can take a number of forms, which are summarised in figure 8.3. If $(\mathcal{M}_p \mid \mathcal{M}'_r) \wr A \backslash B \subset \mathcal{M}_q$ then clearly the next iteration of $\mathcal{M}'_r$ must include actions and states from **Res** (remember **Res** will be a (disc-connected) machine). We can state this as $\mathcal{M}''_r = \mathcal{M}'_r \oplus \mathbf{Res}$. Likewise, $(\mathcal{M}_p \mid \mathcal{M}'_r) \wr A \backslash B \supset \mathcal{M}_q$, in which case $\mathcal{M}''_r = \mathcal{M}'_r \ominus \mathbf{Res}$. If $(\mathcal{M}_p \mid \mathcal{M}'_r) \wr A \backslash B \not\subset \mathcal{M}_q$ and $(\mathcal{M}_p \mid \mathcal{M}'_r) \wr A \backslash B \not\supset \mathcal{M}_q$ then $\mathcal{M}''_r = \mathcal{M}'_r \oslash \mathbf{Res}$. We summarise this be defining a functional **F** as :-

E.1 Definition

$$\mathcal{M}''_r = \mathbf{F}(\mathcal{M}'_r, \mathbf{Res}) = \begin{cases} \mathcal{M}'_r \oplus \mathbf{Res} & \text{if } (\mathcal{M}_p \mid \mathcal{M}'_r) \wr A \backslash B \subset \mathcal{M}_q \\ \mathcal{M}'_r \ominus \mathbf{Res} & \text{if } (\mathcal{M}_p \mid \mathcal{M}'_r) \wr A \backslash B \supset \mathcal{M}_q \\ \mathcal{M}r' \oslash \mathbf{Res} & \text{if } (\mathcal{M}_p \mid \mathcal{M}'_r) \wr A \backslash B \not\subset \mathcal{M}_q \text{ and } (\mathcal{M}_p \mid \mathcal{M}'_r) \wr A \backslash B \not\supset \mathcal{M}_q \end{cases}$$

We could apply Cauchy sequence criteria to define convergence and divergence [78]. Further research is clearly to establish the validity of this concepts. However, both the transformation of observationally equivalent interface equations, and iterative methods to solve it, would seem to present a number of challenges.

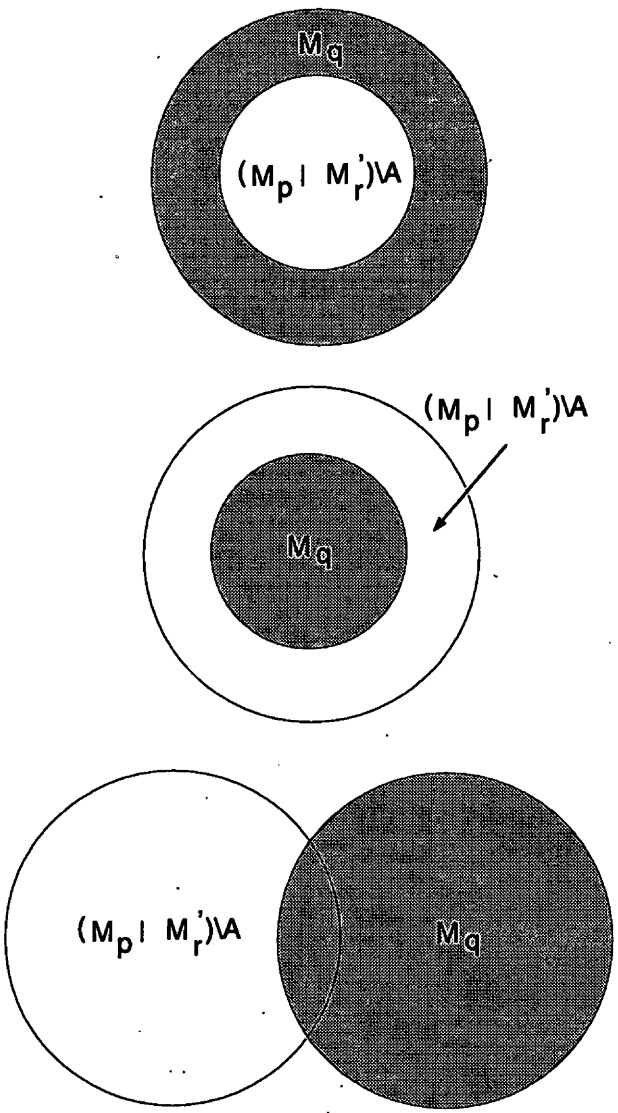


Figure 8.3: The mechanics of the iterative process.

F Quotient machines and generalised automata theory

This flowchart is based on a chart developed by Arbib and Manes in [4]. We include have included this in order to show the deeper links between automata theory and other disciplines — some of which are not immediately obvious. It puts in context the notion of generalised machines. Of particular interest here is the notion of quotient machine. The author has used this diagram to place the research of this thesis within the context of generalised automata theory. Italics and dashed lines indicate additions resulting from this thesis. That the work fits such a schema is encouraging, and very suggestive of the deeper links alluded to in this thesis.

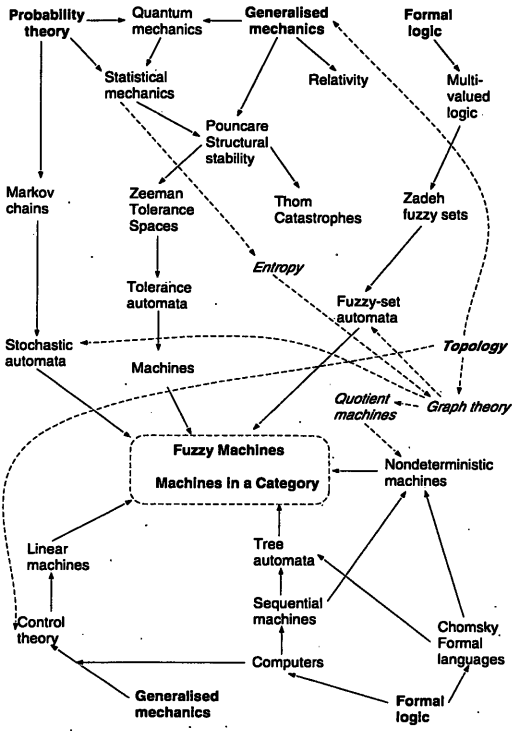


Figure 8.4: The research in the context of generalised automata theory.